HOCHSCHULE
MITTWEIDA
UNIVERSITY OF
APPLIED SCIENCES

BACHELORARBEIT

Herr
Andreas Siegl

Performance-Analyse von Lokalisierungsdiensten auf der iOS Plattform am Beispiel einer standortbasierten Zeiterfassung

2015

Fakultät **Angewandte Computer- und Biowissenschaften**

BACHELORARBEIT

Performance-Analyse von Lokalisierungsdiensten auf der iOS Plattform am Beispiel einer standortbasierten Zeiterfassung

Autor:
Andreas Siegl

Studiengang:
Medieninformatik und Interaktives Entertainment

Seminargruppe:
Mi12w2-B

Erstprüfer:
Prof. Dr.- Ing. Wilfried Schubert

Zweitprüfer:
Dipl.- Ing Bastian Buder

Mittweida, 11 2015

Bibliografische Angaben

Siegl, Andreas: Performance-Analyse von Lokalisierungsdiensten auf der iOS Plattform am Beispiel einer standortbasierten Zeiterfassung, 43 Seiten, 20 Abbildungen, Hochschule Mittweida, University of Applied Sciences, Fakultät Angewandte Computer- und Biowissenschaften

Bachelorarbeit, 2015

Referat

In der vorliegenden Bachelorarbeit wird die Eignung eines Smartphones, spezieller eines iPhones, für die standortbasierte Zeiterfassung untersucht. Betrachtet werden technische Ansätze und die auf der iOS-Plattform verfügbaren Verfahren iBeacon und Geofencing. Dabei soll mittels einer Präsenzdetektion ermittelt werden, ob sich ein Mitarbeiter in der Firma oder am Arbeitsplatz aufhält und dementsprechend eine Zeiterfassung gestartet werden. Die Verfahren werden hinsichtlich ihrer Eignung im Bezug auf aufgestellte Anforderungen und auf ihre reaktive sowie proaktive Verwendung beurteilt.

Schwerpunkte

- Untersuchung verfügbarer technischer Ansätze auf der iOS - Plattform und Vergleich ihrer Eigenschaften
- Implementierung eines Prototypen mit mindestens zwei verschiedenen Ansätzen
- Analyse der Performance in der Praxis insbesondere hinsichtlich der betriebssystemseitigen Einschränkungen (z.B. Aktualisierungsintervall)

I. Inhaltsverzeichnis

II. Abbildungsverzeichnis

III. Tabellenverzeichnis

IV. Abkürzungsverzeichnis

API	application programming interface
GPS	Global Positioning System
IoC	Inversion of Control
MIT	Massachusetts Institute of Technology
NFC	Near Field Communication
QR	Quick Response
RADAR	Radio Aircraft Detection and Ranging
RFID	Radio Frequency Identification
RSSI	Received Signal Strength Indication
SIG	Special Interests Group
SSID	Service Set Identifier
UUID	Universal Unique Identifier

V. Danksagung

Ich möchte mich an dieser Stelle zunächst bei allen bedanken, die mich während der Anfertigung dieser Bachelorarbeit unterstützt haben.

Mein besonderer Dank gilt an dieser Stelle Herrn Prof. Dr.-Ing. Wilfried Schubert sowie Herrn Dipl.-Ing. Bastian Buder die mich, während der Anfertigung dieser Arbeit, betreut und mit kritischem Hinterfragen sowie wertvollen Hinweisen unterstützt haben.

Auch meinen Vorgesetzten und Kollegen der develAPPers UG möchte ich an dieser Stelle für die Hilfe während der Anfertigung danken.

Ein besonderer Dank gilt auch meiner Schwester, der Familie sowie meiner Freundin Claudia für die kontinuierliche Unterstützung während der Bearbeitung dieser Arbeit sowie in der Zeit des Studiums.

1 Einleitung

Zeiterfassung ist in vielen Berufsfeldern ein wichtiger Faktor, welcher sich direkt auf den Erfolg einer Firma auswirken kann. Um Aussagen über den zeitlichen und finanziellen Stand von Projekten treffen zu können und für das Optimieren innerbetrieblicher Abläufe ist es notwendig, die Arbeitszeit der jeweiligen Mitarbeiter zu erfassen, zu verwalten und auszuwerten. Auch die Berechnung des Monatsgehalts der Mitarbeiter kann von der Zeiterfassung abhängig sein. Um die Arbeitszeit von Mitarbeitern zu erfassen, gibt es eine Reihe gängiger Systeme. So existieren beispielsweise Implementierungen, welche mittels Hardware-Terminals an Ein- sowie Ausgängen eine Zeiterfassung durchführen oder aber Mitarbeiter tragen die eigenen Arbeitszeiten selbst in bestimmte Stundenzettel ein. Solche Systeme sind meist wenig nutzerfreundlich, teils fehleranfällig oder spiegeln die erfassten Zeiten ungenau wieder. Eine nutzerfreundlichere Variante wäre eine mobile Arbeitszeiterfassung unter Zuhilfenahme eines Smartphones.

Daraus ergibt sich die Frage: Ist es mittels der in einem Smartphone verbauten Sensorik möglich und sinnvoll die Arbeitszeit von Mitarbeitern zu erfassen ohne dabei Nachteile im Hinblick auf Genauigkeit und Nutzerfreundlichkeit zu riskieren?

In aktuellen Smartphones ist eine Vielzahl an Sensoren verbaut, welche Möglichkeiten zur mobilen Arbeitszeiterfassung bieten können. Es ist zu klären, welche der Möglichkeiten für diesen Anwendungsfall zum Einsatz kommen können und in wie weit sich diese in Bezug auf Nutzerfreundlichkeit sowie Genauigkeit eignen.

In Kapitel 2 wird zuerst auf die Grundlagen von standortbasierten Diensten eingegangen. Es werden dabei einige mögliche Dienste erläutert und der Begriff Ortung geklärt.

Nach der Klärung der Grundlagen folgt in Kapitel 3 anfangs eine Fallstudie zur standortbasierten Zeiterfassung. Hier wird geklärt, welches Einsatzgebiet es dafür gibt und wie sie genutzt werden kann. Anschließend werden Anforderungen an ein solches System, im Bezug auf Genauigkeit, Reaktionszeit und Energiebedarf, aufgestellt. Nach der Klärung der Anforderungen werden die technischen Ansätze geklärt. Dabei findet eine genauere Betrachtung der beiden Hauptverfahren Geofencing und iBeacon statt. Des Weiteren werden alternative Verfahren aufgezeigt und es wird erläutert, weshalb sie für den Anwendungsfall der standortbasierten Zeiterfassung nicht geeignet sind. Nach der Betrachtung der technischen Ansätze wird auf die Plattformuntersützung eingegangen. Zwar beschäftigt sich diese Arbeit hauptsächlich mit den Möglichkeiten der iOS Plattform, jedoch sind die verwendeten Verfahren auch auf anderen Plattformen einsetzbar. Kapitel 3 wird von einem Konzept für die Arbeitszeit-Kontierung innerhalb und außerhalb von Gebäuden abgeschlossen.

Um die Messungen der Genauigkeit durchführen zu können und um die Frage nach einer möglichen Umsetzung zu klären, beschäftigt sich Kapitel 4 mit der Umsetzung einer Prototyp-App unter iOS sowie dem Erstellen zweier Frameworks, welche die Funktionalität von Apples Ortungsdiensten vereinfachen sollen. Hierbei wird grob auf den Aufbau der Applikation eingegangen und es werden etwaige Design-Entscheidungen begründet.

Kapitel 5 beschäftigt sich mit der Performanceanalyse der beiden verwendeten Verfahren Geofencing und iBeacon. Dabei wird jeweils der Versuchsaufbau und die erfassten Werte erläutert. Anschließend wird näher auf mögliche Einflussfaktoren eingegangen. Abschließend findet eine Auswertung der beiden Messreihen statt.

2 Grundlagen

In diesem Kapitel werden die Grundlagen für eine standortbasierte Zeiterfassung beschrieben. Genauer wird dabei auf die Unterteilung von Ortungsverfahren und mögliche Technologien zur Standortbestimmung eingegangen. Hauptaugenmerk liegt dabei auf Technologien, welche auf Smartphones zum Einsatz kommen können.

2.1 Ortungsverfahren

Zur Bestimmung der aktuellen Position gibt es mehrere Verfahren. Diese lassen sich in die nachfolgenden zwei Hauptgruppen untergliedern.

Eigenortung
Die Ortung eines Objektes findet ausschließlich mit eigenen Mitteln statt. Ein Beispiel hierfür ist die Ortsbestimmung durch Höhenwinkelmessung, welche früher zur Navigation auf See eingesetzt wurde. Neben dieser sogenannten autonomen Eigenortung gibt es noch das System der kooperativen Eigenortung. Hierbei wird die Position unter Zuhilfenahme eines externen Systems, welches bestimmte Informationen bereitstellt, ermittelt. Hier können das amerikanische GPS oder das europäische Galileo benannt werden. Zur Bestimmung müssen Signale von mindestens vier Satelliten empfangen werden. Diese senden ihre aktuelle Position und die genaue Uhrzeit. Aus den durch das Senden resultierenden Signallaufzeiten kann so ein GPS-Empfänger die eigene aktuelle Position ermitteln [Mas93].

Fremdortung
Bei der autonomen Fremdortung wird die Ortung des Objektes von einer externen Instanz durchgeführt und verläuft ohne eigene Hilfe [Hic09, Seite 4]. Das RADAR-Verfahren ist ein Beispiel für Fremdortung, da bei der Ortung lediglich elektromagnetische Wellen im Radiofrequenzbereich, beispielsweise von einem Flugzeugrumpf, reflektiert werden. Die Echos werden danach vom Empfänger interpretiert und in Bezug auf Entfernung, Geschwindigkeit und Position ausgewertet [Mah98, Seite 33]. Eine Variante ist, wie bei Eigenortung, die kooperative Fremdortung. Hierbei wird das Ortungssystem aktiv vom zu ortenden Objekt mit Informationen unterstützt.

Die beiden in dieser Arbeit beschriebenen Verfahren, iBeacon und Geofencing, sind dabei unter kooperative Eigenortung einzuordnen, da beim Geofencing eine Unterstützung durch GPS, WLAN und Funkzellen-Positionen erfolgt. Die Positionsbestimmung

oder die Präsenzdetektion mittels iBeacon wird von den gleichnamigen Geräten unterstützt.

2.2 Geofencing

Als Geofencing wird die Möglichkeit bezeichnet mittels eines satellitengestützten Systems, wie GPS, ein bestimmtes Gebiet örtlich einzugrenzen oder einzuzäunen (fence: engl. für Zaun). Ein solcher Bereich kann auf einem mobilen Endgerät mittels Breitengrad (Latitude) und Längengrad (Longitude) vordefiniert werden. Das Betreten oder Verlassen des eingegrenzten Bereiches kann als Ereignis registriert und für aufbauende Funktionen, wie etwa Benachrichtigungen durch das Smartphone, ausgewertet werden. Ein solcher Bereich kann in Kreisform mit bestimmtem Radius, als Rechteck oder auch

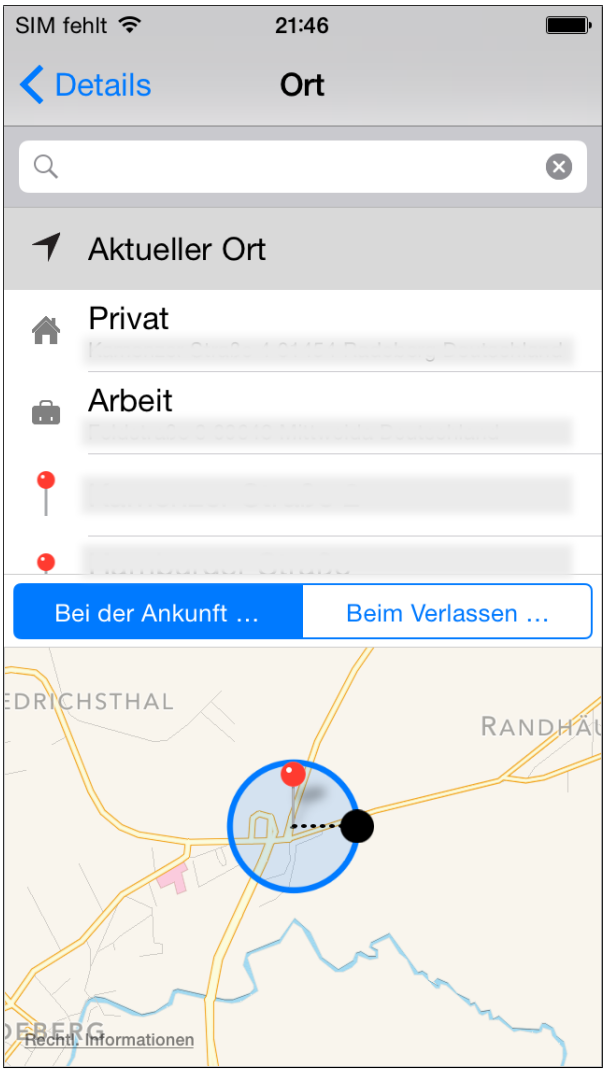


Abbildung 2.1: Geofencing in Apples „Erinnerungen“-App[1]

[1] Aufgenommen auf einem iPhone 5, iOS 8.4.1

komplex als Polygonzug mit einem „Innen“ und einem „Außen“ definiert werden. Auf diese Funktionalität baut beispielsweise Apples „Erinnerungen“-App auf. In dem Systemtool, welches auf jedem iOS und OS X Gerät vorinstalliert ist, können Erinnerungen angelegt werden, welche beim Betreten oder Verlassen eines bestimmten Bereiches angezeigt werden. Apple bietet hierbei nur die Einzäunung mittels eines Radiuses von minimal 100m und maximal 2.414.016m an[2]. Die Standortbestimmung greift unter iOS mittlerweile nicht mehr ausschließlich auf das GPS Signal zurück. Der aktuelle Standort wird über eine Kombination aus GPS, WLAN- und GSM- Ortung errechnet. Für die WLAN- und GSM-Ortung betreibt Apple eine sogenannte „crowd-sourced“-Datenbank. iOS-Geräte senden hierfür bei aktivierter Option „Ortungsdienste“ in regelmäßigen Abständen per Geotagging markierte Positionen von WLAN-Hotspots und Mobilfunkmasten[3] an Apple. Somit ist eine Ortung nicht ausschließlich von GPS abhängig.

2.3 iBeacon

iBeacon ist ein von Apple im Jahr 2013 eingeführter proprietärer Standard zur Navigation und Präsenzdetektion innerhalb von Räumen und Gebäuden. Beacons (engl. für Leuchtfeuer) sind kleine Platinen, welche mit einem Bluetooth Low Energy (BLE oder Bluetooth Smart[4]) Modul ausgestattet sind und oft von einer Knopfzelle betrieben werden können. Hierbei kann eine erhebliche Laufzeit von bis zu drei Jahren erreicht wer-

Abbildung 2.2: Logo Apple iBeacon[5]

den[6]. Sie senden in bestimmten Intervallen einen sogenannte Proximity UUID, einen

[2] Getestet auf einem iPhone 6 unter iOS 8.4.1
[3] https://support.apple.com/de-de/HT203033 [letzter Zugriff: 17.09.2015]
[4] http://www.bluetooth.com/Pages/Bluetooth-Smart.aspx [letzter Zugriff: 31.08.2015]
[5] https://developer.apple.com/ibeacon/ [letzter Zugriff: 15.09.2015]
[6] http://developer.estimote.com [letzter Zugriff: 31.08.2015]

Major- und einen Minor-Wert. Hierbei umfasst die UUID 16 Bytes an Daten, die beiden Werte Major und Minor jeweils 2 Byte. Erkannt werden können solche Leuchtfeuer von allen Geräten, die den Bluetooth LE Standard unterstützen. Eine umfängliche Liste an unterstützen Geräten ist bei der Bluetooth SIG abrufbar[7]. Eine Positionsbestimmung von mobilen Endgeräten kann in einem Raum mit mindestens drei Beacons erfolgen. Hierbei wird die Signalstärke gemessen und mit einem Standardwert, welcher sich auf 1m Entfernung bezieht, verglichen. So kann mittels sogenannter Trilateration die Position im zweidimensionalen Raum bestimmt werden. Der Abstand zu einem iBeacon wird von Apple in vier verschiedene Kategorien unterteilt: Unknown, Immediate, Near und Far. Unknown (unbekannt) bedeutet hierbei, dass die Signalstärke nicht gemessen werden kann oder zu gering ist. Dies ist der Fall, wenn eben mit dem Monitoring, also dem Überwachen der Region, begonnen wurde oder das Gerät den Radius des Beacons verlassen hat. Immediate (unmittelbar) bedeutet, dass sich das mobile Gerät in unmittelbarer Nähe des Beacons befindet. Near (Nah) beschreibt eine Distanz von 1-3m Radius um das Beacon. Far (Fern) bedeutet, dass das Beacon zwar wahrgenommen wird, sich aber zu weit entfernt befindet, um eine genaue Aussage über die Entfernung in Metern treffen zu können[8]. Für eine Präsenzdetektion ist nur ein iBeacon notwendig.

2.4 RFID

RFID ist eine vielfältig eingesetzte Technologie zur automatischen Identifikation. Es besteht eine Ähnlichkeit zum Barcode verfahren in Handel und Versand. Während beim Barcode-Verfahren ein Laserscanner und ein optischer Code verwendet wird, basiert RFID hingegen auf einem Lesegerät mit einer Spule und sogenannten RFID-Transpondern, welche einen einen Mikrochip und ebenfalls eine Spule besitzen. Durch Übermittlung der benötigten Energie für den Betrieb, Daten und Takt durch das Lesegerät an den Transponder wird dieser aktiviert und sendet so die Antwortdaten zurück. Der Vorteil von RFID gegenüber herkömmlichen Barcode Systemen ist, dass keine direkte Sichtverbindung zwischen Transponder (Tag) und Lesegerät bestehen muss. Des Weiteren können mehrere Tag´s gleichzeitig gescannt werden [TT10, Seite 2]. Für die automatische Identifikation wird immer ein RFID-Lesegerät und ein RFID-Tag benötigt, wobei das Lesegerät mit der Kommunikation nach dem Frage-Antwort-Prinzip beginnt [LR10]. Wie in [Mat11] beschrieben, kann RFID auch für eine Lösung zur Arbeitszeiterfassung eingesetzt werden. Dazu kann sich ein Mitarbeiter mittels eines RFID-Tag's, beispielsweise einem Schlüsselanhänger, an einem Zeiterfassungssystem anmelden.

[7] http://www.bluetooth.com/Pages/Bluetooth-Smart-Devices-List.aspx [letzter Zugriff: 19.08.2015]
[8] https://developer.apple.com/ibeacon/Getting-Started-with-iBeacon.pdf [letzter Zugriff: 31.08.2015]

2.5 NFC

NFC ist ein auf RFID (siehe 2.3 RFID) basierender Standard zur Datenübertragung über kurze Distanzen. NFC arbeitet im Frequenzbereich von 13,56 MHz [LR10, Seite 13] und bietet die Möglichkeit Daten mit bis zu 424 kbits/s über eine Entfernung von maximal 10cm zu übertragen [LR10, Seite 38]. NFC ist kompatibel zu bestehenden RFID-Standards in diesem Frequenzbereich. Im Gegensatz zur RFID-Technologie ist ein Gerät, welches NFC unterstützt, in der Lage NFC-Tag's sowohl zu Scannen als auch selbst als Transponder zu fungieren (vgl. Abbildung 2.3). Aktuell wird NFC beispielsweise für das kontaktlose Bezahlen mittels Kreditkarte und für Dienste wie Apple Pay[9] verwendet.

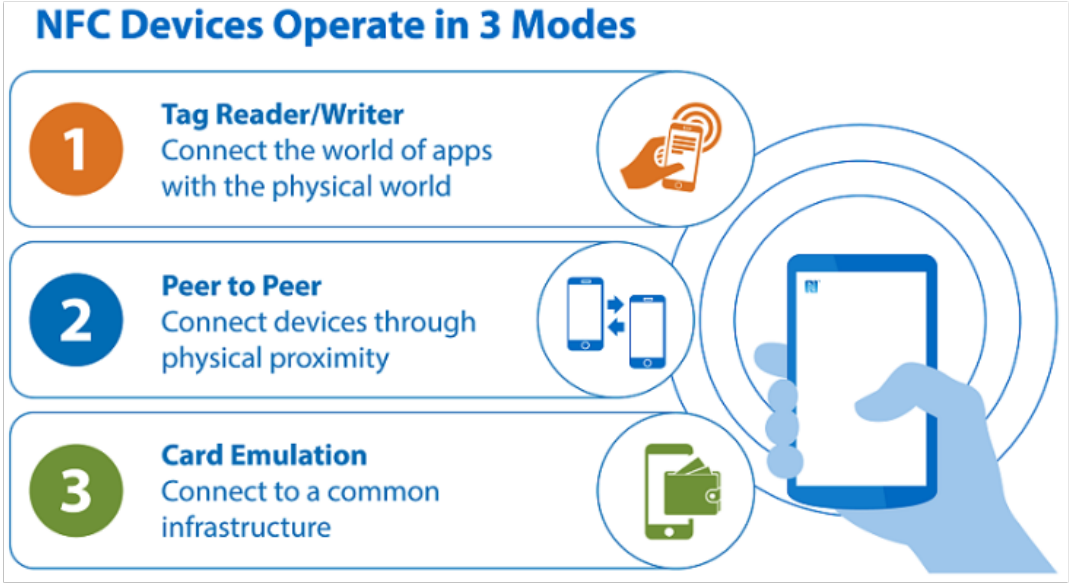


Abbildung 2.3: Darstellung der einzelnen NFC-Modi[10]

[9] Zur Zeit nur in den USA und England verfügbar
[10] Bildquelle: http://nfc-forum.org/what-is-nfc/what-it-does/ [letzter Zugriff: 23.08.2015]

3 Standortbasierte Zeiterfassung

Dieses Kapitel befasst sich zunächst mit dem Erstellen einer Fallstudie zum Thema standortbasierte Zeiterfassung. Hierbei sollen mögliche Anwendungsszenarien genauer beschrieben werden. Anschließend werden Anforderungen an ein System, im konkreten Fall an eine App, aufgestellt. Dabei findet eine genauere Betrachtung der Themen Reaktionszeit, Genauigkeit sowie allgemeine Verfügbarkeit und Energieeffizienz statt. Weiterhin werden die Eigenschaften der Verfahren iBeacon und Geofencing betrachtet und alternative Verfahren zur Ortung aufgezeigt.

3.1 Fallstudie

In vielen größeren und kleineren Betrieben wird die Arbeitszeit der Mitarbeiter bereits durch Systeme gemessen, welche eine automatisierte Auswertung und einfache Optimierung der innerbetrieblichen Abläufe ermöglichen. Meist sind solche Systeme mittels Terminals an Ein- und Ausgangsbereichen realisiert. Beim Betreten des Arbeitsplatzes wird beispielsweise eine Magnetkarte genutzt, welche den Mitarbeiter identifiziert, um ihn so am System anzumelden. Der selbe Vorgang erfolgt beim Verlassen des Arbeitsplatzes. Durch ein solches System lässt sich die Arbeitszeit, zum Beispiel von Außendienstmitarbeitern oder Pflegekräften, welche zu Hausbesuchen oder einem Kunden fahren, nur schwer ermitteln. Für diese Personen ist eine manuelle Erfassung der Arbeitszeiten notwendig (vgl. [Mat11]). Ebenso sind Zeiterfassungen bei denen der Nutzer Zeiteinträge manuell anlegt, beispielsweise der Web-Dienst „Toggl“[11], fehleranfällig falls der Nutzer vergisst seine Arbeitszeit zu erfassen. Sofern sie nicht schon digital vorliegen, müssen solche Daten zu einem späteren Zeitpunkt von einem Verwaltungsapparat in eine digitale Form überführt oder überarbeitet werden. Der Aufwand des Erfassens und Übertragens der gesammelten Daten ist einerseits immer wiederkehrend und zeitaufwändig, andererseits treten leicht Fehler bei der initialen Aufnahme und der späteren Übertragung in ein anderes System auf. Eine Auswertung sowie die Optimierung von Projekten und Prozessen ist damit nur stark verzögert möglich.
Abhilfe kann ein automatisiertes, standortbasiertes Zeiterfassungssystem bieten. So könnte unter Zuhilfenahme von Smartphones und deren Sensoren ein System entwickelt werden, welches die Zeiterfassung stark vereinfacht oder sogar komplett automatisiert. Die Zeiterfassung kann beim Eintreffen automatisch gestartet und beim Verlassen des Arbeitsplatzes automatisch gestoppt werden.

[11] `https://toggl.com` [letzter Zugriff: 13.11.2015]

3.2 Anforderungen

Um eine möglichst genaue Erfassung der Arbeitszeit erreichen zu können, muss ein solches System verschiedene Anforderungen erfüllen. In diesem Abschnitt sollen die Anforderungen erläutert und ihre Wichtigkeit bewertet werden.

Reaktionszeit/Verfügbarkeit
Für die Aufzeichnung von wichtigen Daten ist eine schnelle Reaktionszeit unerlässlich. Wenn die Aufzeichnung der Arbeitszeit bei Betreten des Arbeitsplatzes beginnen soll, muss darauf geachtet werden, dass eben diese Aufzeichnung ohne größere Latenzzeiten abläuft. Die Aufzeichnung muss also möglichst zeitnah bei Betreten oder Verlassen starten beziehungsweise stoppen. Eine zu hohe Reaktionszeit hätte verfälschte Einträge als Resultat. Diese wären im schlimmsten Fall für eine Auswertung unbrauchbar. Ebenso sollte die Zeiterfassung auch dann möglich sein, wenn kein GPS und/oder Internet vorhanden ist. Für den Anwendungsfall einer Arbeitszeiterfassung erscheint eine minutengenaue Erfassung als ausreichend. Abweichungen im Sekunden-Bereich können an dieser Stelle hingenommen werden.

Örtliche Genauigkeit
Ein System, welches standortabhängig Arbeitszeiten erfasst, hat hohe Anforderungen an die örtliche Genauigkeit. Die Zeiterfassung darf, beispielsweise in einem Büro, noch nicht beginnen, wenn man sich noch vor dem Gebäude aufhält. Ebenso muss das Abschließen eines Zeiteintrages direkt beim Verlassen des Arbeitsplatzes stattfinden. Eine nachträgliche Bearbeitung der erfassten Zeiten kann zulässig, sollte jedoch nicht erforderlich sein. Um eine hinreichend genaue Erfassung zu erreichen, ist für den Innenbereich eine auf wenige Meter genaue Erfassung anzustreben. Außerhalb von Gebäuden, bspw. auf einem Firmengelände, hängt die Genauigkeit stark von der örtlichen Ausdehnung des Gebietes ab.

Energiebedarf
Ein kontinuierliches Aktualisieren der Position eines Endgerätes mittels GPS kann unter umständen einen großen Einfluss auf die Akkulaufzeit haben. Um die aktuelle Position abzufragen sollte dementsprechend auf die Frequenz der Aktualisierung geachtet werden. Je öfter der Standort angefragt wird, umso öfter wird auch das GPS Modul aktiviert [App14b, vgl. S.12]. Es sollte daher darauf geachtet werden, die Position nur bei signifikanten Änderungen zu aktualisieren. Bei einer Präsenzdetektion mittels iBeacon ist die Auswirkung auf den Akku hingegen als eher gering einzuschätzen. Wie in Kapitel 2.3 beschrieben nutzt diese Technik den Bluetooth Smart/LE Standard, welcher nur einen sehr geringen Energiebedarf aufweist. Es ist ein Energiebedarf anzustreben, mit

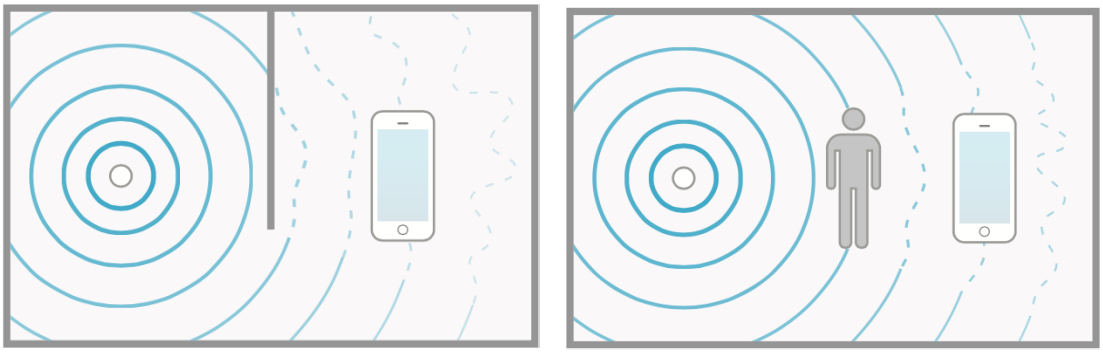
welchem gewährleistet werden kann, dass das Endgerät bei einer normalen Nutzung einen Arbeitstag lang verfügbar ist.

Infrastruktur
Der Bedarf an zusätzlicher, komplexer Hardware ist mit einem erhöhtem Installations- und Wartungsaufwand verbunden. Um den Installationsaufwand sowie die Kosten eines solchen Systems so gering wie möglich zu halten, soll weitestgehend autark von zusätzlicher Hardware gearbeitet werden.

3.3 Technische Ansätze

3.3.1 iBeacon

Durch den Bedarf an zusätzlicher Hardware und der im Gegensatz zu Geofencing geringen Reichweite eines Beacons, kann die Eignung größtenteils auf den Innenbereich von Gebäuden reduziert werden. Ein Einsatz im Freien ist zwar denkbar und möglich, jedoch stark vom Standort und der räumlichen Ausdehnung des Arbeitsbereiches abhängig. Wie in Kapitel 2.3 beschrieben, hat ein Beacon eine begrenzte Reichweite und ist daher im Außenbereich nur über relativ kurze Distanzen verwendbar. Die Beacons der Firma Estimote weisen laut Spezifikation beispielsweise eine Reichweite von 40-50 Metern auf[12]. Bei einer Nutzung innerhalb von Gebäuden, beispielsweise Büroräumen, ist weiterhin zu bedenken, dass die Bluetooth-Signale von Wänden und auch Personen gedämpft werden, wodurch sich die Signalstärke und damit die Reichweite erheblich reduziert.



(a) Dämpfung einer Wand (b) Dämpfung einer Person

Abbildung 3.1: Darstellung äußerer Einflüsse auf die Ausbreitung von Bluetooth-Signalen aus [App14a]

[12] http://developer.estimote.com [letzter Zugriff: 09.10.2015]

3.3.2 Geofencing

Das Geofencing-Verfahren erfordert im Gegensatz zu iBeacons keine zusätzliche Hardware-Infrastruktur, da die Ortung größtenteils auf das bereits bestehende GPS aufsetzt (siehe: 2.2 Geofencing). Die Eignung für die positionsabhängige Zeiterfassung ist hauptsächlich im Außenbereich zu finden. Durch die teilweise Abhängigkeit eines GPS-Signals ist eine Ortung innerhalb von Gebäuden zwar möglich, kann jedoch ungenau und fehlerbehaftet sein. Unter iOS wird die Genauigkeit der aktuellen Position in der „Karten"-Anwendung als heller, blauer Kreis dargestellt. Ein kleiner Kreis um die aktuelle Position bedeutet dabei eine hohe Genauigkeit. Ein großer Kreis bedeutet geringere Genauigkeit. Hier ist zu beobachten, dass die Genauigkeit innerhalb von Gebäuden als sehr gering dargestellt wird und sich der ermittelte Standort oft ändert.

3.3.3 Alternative Verfahren

Neben der in 3.3.1 und 3.3.2 beschriebenen Verfahren, gibt es noch weitere technische Möglichkeiten, welche nachfolgend genannt werden sollen.

Mittels installierter WLAN Hotspots kann ebenfalls eine Präsenzdetektion stattfinden. Ein Gerät, welches sich innerhalb des Bereiches eines solchen Funknetzwerkes befindet, kann mittels der RSSI einen ungefähren Abstand zum Ursprung des Netzwerkes berechnen [HB14]. Die Identifikation des Netzwerkes erfolgt dabei über den SSID des Netzwerkes. Der Nachteil ist jedoch, dass es von Seiten des Betriebssystems iOS keine öffentliche Schnittstelle für solche Messungen gibt. Zwar kann innerhalb einer laufenden App die SSID des aktuellen Netzwerks ermittelt werden, jedoch gibt es keine Möglichkeit den Wechsel zu einem anderen Netzwerk als Event zu registrieren. Des Weiteren ist der SSID keinesfalls ein eindeutiges Merkmal wie man an dem in vielen Hochschulen verfügbarem Netz „eduroam" sehen kann. Alternativ könnte, mittels kooperativer Fremdortung, eine Auswertung stattfinden, welche Mitarbeiter im WLAN-Netz eingebucht sind. Ein solches Verfahren würde jedoch ein Eingreifen in bereits bestehende Infrastruktur voraussetzten.

Wie in [HZH+12] beschrieben, ist eine Indoor-Ortung auch durch akustische Signale möglich. Bei diesem Verfahren sendet ein zu ortendes Mobilgerät einen für den Menschen nicht wahrnehmbaren Ton aus, welcher von Audioempfängern, die in dem betreffenden Raum installiert sein müssen, aufgenommen wird. In einem betreffenden Raum müssen für die genaue Positionsbestimmung mindestens 3 dieser Empfänger installiert sein, welche zeitlich exakt synchronisiert sein müssen. Anhand der Zeitstempel kann so eine zweidimensionale Position des Mobilgerätes bestimmt werden. Für eine einfache Präsenzdetektion würde allerdings ein einzelner Audio-Empfänger ausreichen. Auch von diesem Verfahren wurde abgesehen, da ebenfalls keine öffentlichen Programmierschnittstellen vorhanden sind und die benötigte Hardware keine Standard-

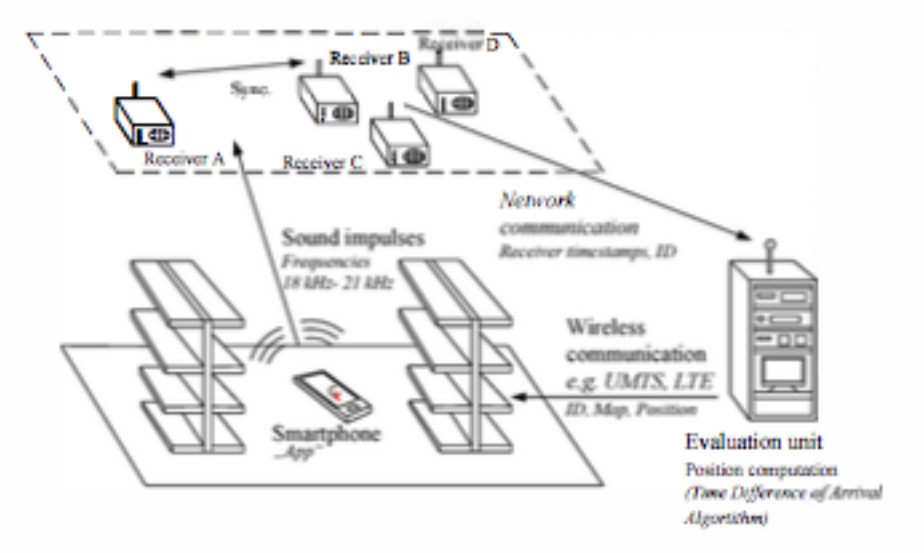


Abbildung 3.2: Darstellung eines akustischen Ortungssystems aus [HZH+12]

Hardware ist.

Eine Ortung oder Präsenzdetektion ist auch optisch Möglich. In [RTI13] wird ein Verfahren beschrieben, bei welchem die Kamera eines Smartphones genutzt wird um QR-Codes, welche sich auf dem Boden befinden, zu scannen (Abbildung: 3.3). Auf einem vorher erstellten Gebäudeplan sind die Positionen der einzelnen Codes mit Position gespeichert. Anhand dieser Informationen kann der Nutzer dann auf einer Karte, welche über den Gebäudeplan gelegt wird, die eigene Position ablesen. Dieses Verfahren ist in den Aspekten Genauigkeit, Instandhaltung und Infrastruktur zwar sehr effizient, jedoch ist dieses System stark von der im Smartphone verbauten Kamera abhängig und das Gerät muss immer in der Hand gehalten werden. Durch die umständliche Nutzbarkeit für den Anwender ist dieses Verfahren ebenfalls nicht für den unter 3.1 beschriebenen Anwendungsfall geeignet.

3.4 Plattformunterstützung

In diesem Abschnitt soll die Plattformunterstützung der beiden verwendeten Verfahren geklärt werden. Dabei werden nur die relevanten Plattformen iOS, Android und die Universal Windows Plattform betrachtet, da der Anteil anderer Betriebssysteme verschwindend gering ist [IDC15].

Abbildung 3.3: Verwendung eines optischen Ortungsverfahrens aus [RTI13] mittels QR

3.4.1 iOS

Auf der iOS Plattform bietet Apple für die Ermittlung der aktuellen Position mehrere Möglichkeiten an. Nicht alle Möglichkeiten sind hier für Programmierer frei zugänglich und/oder konfigurierbar. Seit dem erscheinen des iPhone 6 ist der standard NFC (siehe Kapitel 2.5 NFC) in neuen iPhones und der Apple Watch integriert. Dieser wird jedoch zur Zeit nur für das mobile Bezahlen mittels Apple Pay genutzt. Eine öffentliche Programmierschnittstelle gibt es hierfür noch nicht. Ein direkter Zugriff auf die Ortung über ein WLAN-Netzwerk ist ebenfalls nicht öffentlich zugänglich. Apple verbessert die Ortung im Allgemeinen mittels einer Datenbank von WLAN-Netzwerken und Mobilfunkmasten, um so eine möglichst genaue Ortung auch dann zu gewährleisten, wenn kein oder ein schlechtes GPS Signal verfügbar ist (siehe Kapitel 2.2 Geofencing).

Die gewünschte Genauigkeit kann durch verschiedene Abstufungen vom Entwickler festgelegt werden. Dabei ist zu beachten, dass, je höher die Genauigkeit gewählt wird, auch der Energiebedarf der Anwendung steigt. Die API des CoreLocation Frameworks ermöglicht keine Rückschlüsse auf die zur Ortsbestimmung genutzten Informationsquellen. Es ist lediglich möglich die Ortung auf GPS zu beschränken, in dem ein spezieller Wert in die Konfigurationsdatei der jeweiligen App eingetragen wird.

Abbildung 3.4: Apples Beschreibung wie Ortungsdienste unter iOS verbessert werden[13]

Für die Lokalisierung muss auf Geofencing und/oder iBeacon zurückgegriffen werden. Es ist unter iOS möglich bis zu 20 Geofences pro App zur gleichen Zeit zu überwachen. Für eine Standorterfassung mittels iBeacon muss zusätzlich zu einem unterstützen iOS Gerät ein Beacon erworben werden. Diese gibt es mittlerweile von einer größeren Zahl von Herstellern. Eine Unterscheidung, welche Geräte von welcher Firma hierbei die bessere Wahl darstellen, soll in dieser Arbeit allerdings nicht getroffen werden.

3.4.2 Android

Auf der Android-Plattform ist Geofencing, wie auch unter iOS, als Systemdienst verfügbar. Auf dem Betriebssystem ist es möglich bis zu 100 Geofences pro Nutzer anzulegen, unter iOS sind es hingegen 20 per App. Systemseitig werden benötigte Berechtigungen, wie etwa Ortung, unter Android bei der Installation der App abgefragt. Demnach ist es nicht notwendig die Genehmigung des Nutzers zur Laufzeit einzuholen. Im Gegensatz zu iOS kann ein Deaktivieren der Location Services oder ein Neustarten des Gerätes dazu führen, dass Geofences nicht mehr überwacht werden[14]. Um dieses Verhalten zu umgehen, müsste ein Listener implementiert werden, welcher beim Systemstart die angelegten Geofences neu registriert.

Ebenfalls ist das Ranging, also die Abstandsmessung, sowie das Monitoring, die reine Präsenzdetektion, von Beacons möglich. Google stellt hierfür einerseits den quelloffenen Standard Eddystone[15] zur Verfügung, andererseits ist es auch möglich die Standards iBeacon oder AltBeacon[16] zu verwenden.

Beide Standards, iBeacon und Eddystone, werden dabei beispielsweise von dem in

[13] Aufgenommen auf einem iPhone 6 unter iOS 8.4.1
[14] `http://www.raywenderlich.com/103540/geofences-googleapiclient` [letzer Zugriff: 30.10.2015]
[15] `https://github.com/google/eddystone` [letzer Zugriff: 30.10.2015]
[16] `http://altbeacon.org` [letzter Zugriff: 17.08.2015]

3.3.1 erwähnten Hersteller Estimote unterstützt, welcher jedoch nur für iOS ein passendes SDK bereitstellt. Android bietet für die Verwendung von Beacons ähnliche Funktionen wie iOS, so ist es ebenfalls möglich die Nähe zu einem Beacon über die Signalstärke zu ermitteln sowie sich beim Betreten und dem Verlassen einer Region benachrichtigen zu lassen. Der Standard Eddystone bietet jedoch im Gegensatz zu iBeacon noch einige zusätzliche Funktionen, so ist es beispielsweise möglich zusätzlich zu UUID, Major und Minor weitere Daten wie etwa ein Bild[17] oder eine URL[18] zu übermitteln.

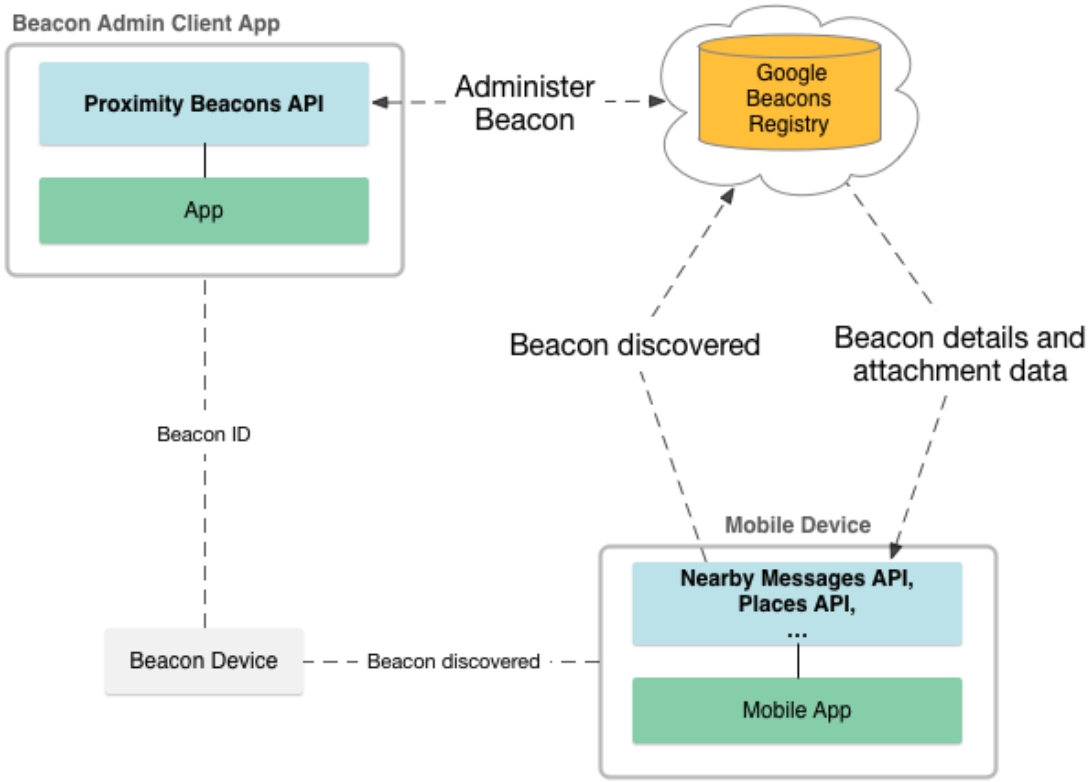


Abbildung 3.5: Darstellung der Google Beacon Komponenten[19]

3.4.3 Windows Phone

Mittels der „Universal Beacon Library“[20], welche über den „nuget“-Paket Manager eingebunden werden kann, ist es auf der Universal Windows Platform ebenfalls möglich, die Standards iBeacon und Eddystone zu verwenden. Dabei unterstützt das Framework auch die zusätzlichen Funktionen des Eddystone-Standards. Es ist ebenfalls möglich Bluetooth-Events zu verarbeiten, wenn sich die App im Hintergrund befindet.

[17] `http://android-developers.blogspot.de/2015/07/lighting-way-with-ble-beacons.html` [letzter Zugriff: 29.08.2015]
[18] `https://github.com/google/eddystone/blob/master/protocol-specification.md` [letzter Zugriff: 29.08.2015]
[19] `https://developers.google.com/beacons/overview` [letzter Zugriff: 30.08.2015]
[20] `https://github.com/andijakl/universal-beacon` [letzter Zugriff: 12.09.2015]

Wie auch auf den anderen größeren Plattformen ist auch auf Windows Phone das Überwachen von Geofence-Regionen möglich. Diese Funktion ist als Systemdienst verfügbar und benötigt keine Drittanbieter-Frameworks. Im Gegensatz zu iOS beträgt der Mindestradius für einen Geofence 50 Meter. Es stehen Events für den Eintritt und das Verlassen einer Region bereit, welche sowohl im Vordergrund (bei aktiver App) als auch im Hintergrund (die App befindet sich im Standby oder ist nicht gestartet) ausgelöst werden.

3.5 Konzept Standortbasierte Zeiterfassung

Im folgenden Abschnitt wird das Konzept der standortbasierten Zeiterfassung bezogen auf die iOS Platform erläutert. Es erfolgt eine Unterteilung in Indoor, also innerhalb von Gebäuden, und Outdoor, also außerhalb von Gebäuden.

3.5.1 Indoor

Für eine Zeiterfassung innerhalb von Gebäuden und Büroräumen eignet sich, wie in 3.3.1 beschrieben, hauptsächlich das iBeacon-Verfahren. Ein System zur Erfassung der Arbeitszeit der Mitarbeiter mittels iBeacon ist nachfolgend am Beispiel von Büroräumen beschrieben.

Um mittels iBeacons die Arbeitszeit von Arbeitskräften zu erfassen, werden, neben einem Smartphone, zusätzlich ein oder mehrere iBeacons benötigt. Der gesamte Aufbau besteht aus drei Teilen: einem Verwaltungssystem, den iBeacons und einer auf den Smartphones der Mitarbeiter installierten App. Die iBeacons müssen für ein solches System konfigurierbar sein. Das heißt, es muss möglich sein, die UUID, den Major und den Minor Wert anzupassen. Als möglicher Hersteller kann an dieser Stelle die Firma Estimote genannt werden, da Beacons dieser Firma einerseits leicht konfigurierbar (RESTful-Api[21]) und andererseits auch mit dem Google-Standard „Eddystone“ kompatibel sind (siehe: 3.4.2). Die UUID‘s werden bei jedem Beacon auf den selben Wert eingestellt. Dieser UUID-Wert repräsentiert die Firma. Die Unterscheidung der einzelnen Arbeitsplätze erfolgt mittels der Major und Minor Werte. Das Beacon wird in unmittelbarer Nähe des Arbeitsplatzes angebracht, beispielsweise hinter einem Monitor. Der auf den Smartphones der Mitarbeiter installierten App wird dann über das Verwaltungssystem, bspw. nach einem Login oder einer ähnlichen Konfiguration, das zugehörige Beacon für den Mitarbeiter mitgeteilt. Die App registriert dieses und kann von nun an, entweder automatisch oder über eine Notification, die Zeiterfassung für den jeweiligen Mitarbeiter durchführen. Mittels einer Notification würde der Nutzer beim Betreten des Bereiches gefragt, ob die Arbeitszeiterfassung gestartet werden soll. Die erfassten Werte werden anschließend über das Internet oder ein lokales Netzwerk an das Verwaltungssystem

[21] `https://estimote.com` [letzter Zugriff: 31.08.2015]

gesendet, welches diese Auswertet. Es ist dabei möglich jedem Mitarbeiter mehrere Beacons zuzuteilen, sowie ein Beacon für mehrere Mitarbeiter zu verwenden. Dies wäre beispielsweise für Besprechungsräume oder ähnliche Bereiche sinnvoll.

3.5.2 Outdoor

Für den Außenbereich eignet sich, wie in 3.3.2 beschrieben, hauptsächlich die Zeiterfassung mittels Geofencing. Nachfolgend ist daher ein Zeiterfassungssystem beschrieben, welches Geofencing nutzt.

Für die Zeiterfassung mittels Geofencing wird im Gegensatz zum iBeacon-Verfahren nur ein Smartphone und ein Verwaltungssystem benötigt. Auf den Smartphones der Mitarbeiter wird, wie auch beim iBeacon-Verfahren, eine App installiert, welche das Erfassen der Zeiteinträge übernimmt. Diese erhält nach dem Login des Mitarbeiters vom Verwaltungssystem die Daten für die geografische Region, in welcher der Mitarbeiter arbeitet. Die Daten für den Arbeitsbereich müssen vorher im Verwaltungssystem eingetragen werden. Die Region wird von der App gespeichert und mittels des Core Location Frameworks registriert. Dabei kann die Region, wie in 2.2 beschrieben, entweder kreisförmig, rechteckig oder komplex als Polygonzug auftreten. Jeweils beim Betreten und Verlassen erfolgt die Zeiterfassung entweder automatisiert oder auf Abfrage. Der Nutzer erhält dabei eine Notification das der Arbeitsbereich betreten oder verlassen wurde. Die ermittelten Zeiteinträge werden anschließend von der App an das Verwaltungssystem zur weiteren Verarbeitung übermittelt.

4 Prototyp für standortbasierte Zeiterfassung

In diesem Kapitel wird der umgesetzte iOS-Prototyp hinsichtlich der verwendeten Technologien und der Architektur beschrieben. Dabei werden ebenfalls die zum Einsatz gekommenen Drittanbieter-Komponenten genannt und deren Einsatz kurz erläutert. Die Unterteilung findet in diesem Kapitel nach der Basis-App und den beiden erstellten Frameworks statt.

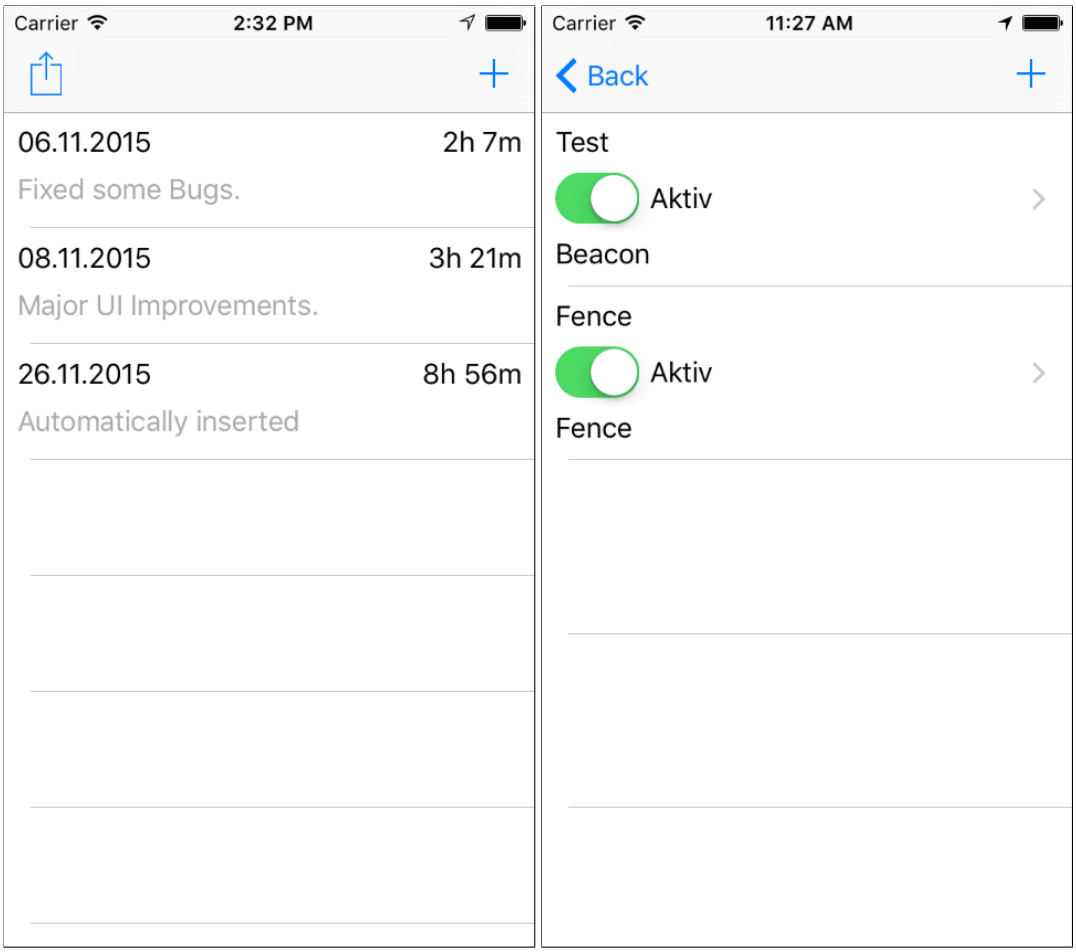
4.1 Basis App

Als Grundgerüst für die standortbasierte Zeiterfassung wurde eine einfache App unter iOS 9 und der Programmiersprache Swift in Version zwei umgesetzt. In der App ist es möglich Zeiteinträge manuell zu erfassen, zu ändern und zu löschen. Ein Zeiteintrag besteht dabei aus einem Startdatum, einem Stopdatum, und einer optionalen Notiz. Intern werden diese Einträge mittels eines UUID verwaltet. Für die Erfassung der Einträge wurde das Framework *Eureka* genutzt[22], welches unter der MIT-Lizenz auf GitHub verfügbar ist. *Eureka* stellt eine einfache Möglichkeit dar, Formulare mit dynamischen Table-Views unter iOS zu erstellen.

Die bereits erfassten Einträge werden in einer Liste (Abbildung: 4.1) mit dem Startdatum, der Dauer und einer Notiz angezeigt und können über jene Liste gegebenenfalls bearbeitet werden. Weiterhin ist es möglich, neue iBeacon- und Geofence-Trigger hinzuzufügen. Diese werden ebenfalls in einer Liste (Abbildung: 4.1(b)) dargestellt und können von hier aus auch editiert werden. Intern werden iBeacons und Geofences als Trigger verwaltet. Ein Trigger besitzt Felder für beide Möglichkeiten, iBeacon sowie Geofence, und wird anhand einer Typ-Eigenschaft unterschieden. Die in der Liste angezeigten Trigger können mittels eines Switches schnell aktiviert sowie deaktiviert werden (Abbildung: 4.1(b)). Für eine einfache Unterscheidung der einzelnen iBeacon und Geofence-Bereiche für den Nutzer besitzt ein Trigger-Objekt noch einen nicht-optionalen Identifikator, welcher bei der Erstellung mit angegeben werden muss.

Um die einzelnen Listen mit den benötigten Daten zu füllen, wird unter iOS eine sogenannte Datasource erstellt. Standardmäßig wird diese bereits von einem verwendeten UITableViewController implementiert. Eine Datasource direkt in einem ViewController zu erstellen, führt jedoch zu überladenem und schlecht wartbarem Quellcode. Des Weiteren wird eine Datasource oft an weiteren Stellen im Programm benötigt. Für eine bessere Code-Qualität wurde daher das Dependency Management (IoC-) Framework

[22] `https://github.com/xmartlabs/Eureka` [letzter Zugriff: 21.10.2015]



(a) Darstellung der einzelnen Zeiteinträge (b) Darstellung der einzelnen Trigger

Abbildung 4.1: Darstellung der Zeiteinträge und Trigger im App-Prototyp

Typhoon[23,24] eingebunden. Dadurch können Abhängigkeiten zentral konfiguriert und ausgetauscht werden. Die in der sogenannten Assembly definierten Abhängigkeiten werden zur Laufzeit aufgelöst.

Für eine einfache Verwaltung von Drittanbieter-Komponenten wurde im Prototyp der Dependency Manager *CocoaPods*[25] eingesetzt. Dieser ist ein Kommandozeilen Programm, welches unter Mac OS installiert werden kann. Mittels CocoaPods ist es möglich Frameworks und Libraries in das Projekt einzubinden. Dazu wird im Hauptverzeichnis ein sogenanntes *Podfile* angelegt in welchem die Abhängigkeiten eingetragen werden können.

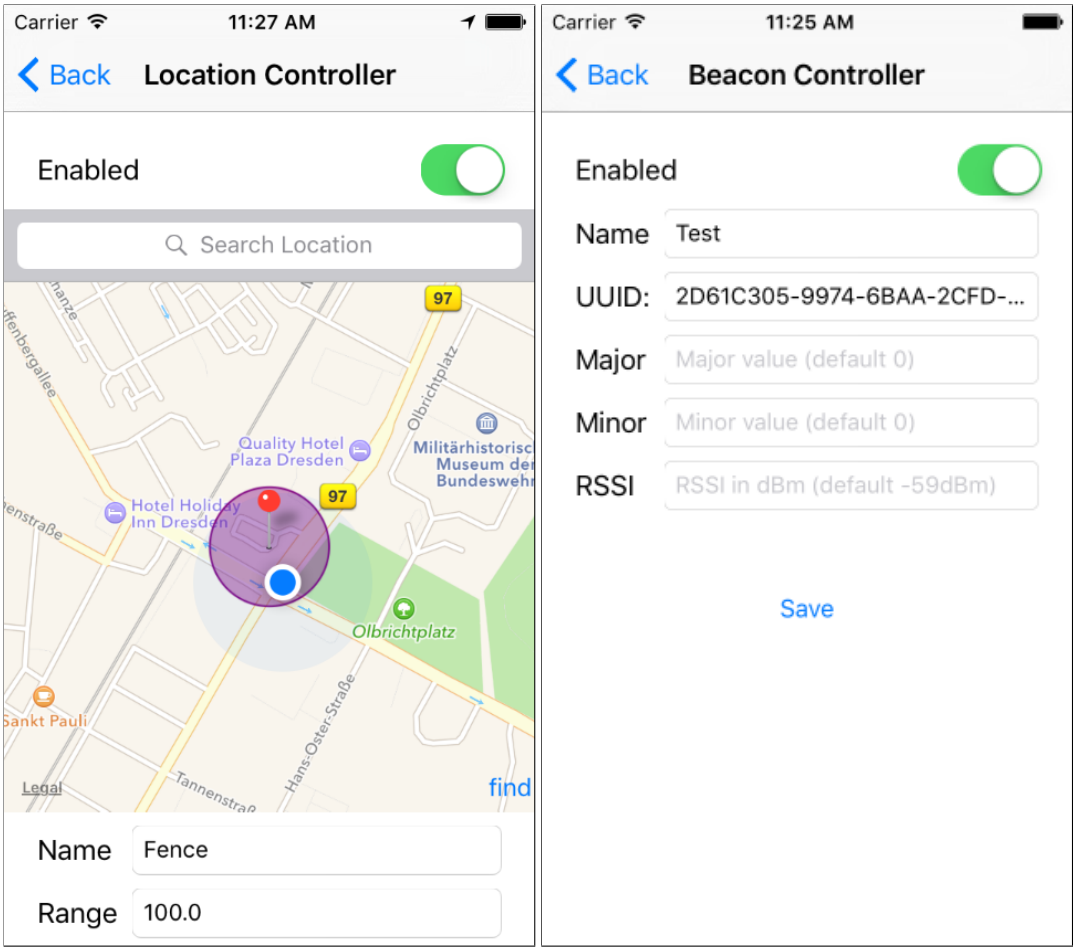
Wie in Listing 4.1 ersichtlich ist, werden die Abhängigkeiten für jedes Target im Projekt einzeln definiert. So ist es auch möglich für eventuelle Unit-Test-Targets spezielle Frameworks einzubinden. Die einzelnen Frameworks/Libraries können mit der Versi-

[23] `https://github.com/appsquickly/Typhoon` [letzter Zugriff: 21.10.2015]
[24] Typhoon befindet sich unter der Apache v2.0 Lizenz
[25] `https://cocoapods.org` [letzter Zugriff: 21.10.2015]

onsnummer angegeben werden. Für den Fall, dass das Framework seit dem einbinden aktualisiert und eventuell Funktionen entfernt oder geändert wurden, kann so die Lauffähigkeit in der aktuellen Version sichergestellt werden.

Listing 4.1: Podfile des App-Prototyps

```
1 platform :ios, '9.0'
2 target 'timesheet' do
3 use_frameworks!
4 pod 'RealmSwift', '~> 0.96.0'
5 pod 'Typhoon', '~>3.3.3'
6 pod 'Eureka', '~>1.1.0'
7 end
```



(a) Darstellung der einzelnen Zeiteinträge (b) Darstellung der einzelnen Trigger

Abbildung 4.2: Anlegen von Triggern im App-Prototyp

Die lokale Persistierung der Einträge erfolgt im Prototyp über das auf iOS, Mac OS und Android verfügbare *Realm*-Framework[26]. *Realm* ist eine mobile Datenbanklösung,

[26] `https://realm.io` [letzter Zugriff: 21.10.2015]

welche unter der Apache 2.0 Lizenz auf GitHub[27] verfügbar ist. Das Ablegen von Objekten erfolgt über ein Model, welches von der Klasse „Object“ erbt. Mit diesem können Schreib- und Lesezugriffe auf den Realm (die Datenbank) ausgeführt werden. Einer der größten Vorteile von *Realm* ist, dass die verwendete Datenbank-Datei auf iOS sowie Android genutzt werden kann.

Listing 4.2: Realm Model Klasse

```
1 class TimeSheet: Object {
2
3     dynamic var startDate = NSDate()
4     dynamic var stopDate: NSDate?
5     dynamic var note: String = ""
6     ...
7 }
```

Eine andere Möglichkeit der Persistierung ist Apples „Core Data“-Framework, jedoch ist Realm für einfache Daten wie Zeiteinträge leichter zu bedienen und verursacht weniger sowie einfacher verständlichen Quellcode.

4.2 Erfassung mittels iBeacon

Für die Erfassung durch iBeacons wurde ein Framework umgesetzt, welches das einfache Anlegen und Verwalten dieser ermöglicht und die Funktionalität des CoreLocation[28]-Frameworks kapselt. Für die Nutzung von iBeacons wird das Core Location Framework von Apple vorausgesetzt, welches die Funktionalität zur Überwachung einer bestimmten Region bereitstellt. Dieses stellt den CLLocationManager[29] und dessen Delegate-Methoden bereit, welche für eine Präsenzdetektion benötigt werden. Das hierbei von Apple verwendete Delegation-Pattern ist bereits vor Swift ein wichtiger Teil der iOS-Programmierung gewesen. Dabei wird ein Objekt (das delegierte) von einem anderen Objekt über auftretende Ereignisse informiert. Dies ist besonders bei asynchronen Aufgaben, wie der Bezug des Standortes, von Nöten.
iBeacons können über die beiden Methoden startMonitoring(:) und stopMonitoring(:) hinzugefügt bzw. entfernt werden. Dazu wurde ein Protocol[30] implementiert, welches gewährleisten soll, dass das übergebene Objekt die nötigen öffentlichen Methoden besitzt. Mit diesen Eigenschaften wird nun ein CLBeaconRegion-Objekt erzeugt. Diese

[27] `https://github.com/realm/` [letzter Zugriff: 21.10.2015]
[28] Ein im iOS-SDK enthaltenes Framework für die Nutzung standortbasierter Dienste
[29] Eine in CoreLocation enthaltene Klasse, welche für die Konfiguration von standortbasierten Events genutzt wird
[30] Ein Protocol ist in der Programmiersprache Swift ähnlich wie ein Interface in Java zu verstehen. Es bezeichnet verschiedene Eigenschaften und Methoden welchen eine Klasse, Struktur oder Enumeration folgen muss.

Region wird nun überwacht und es werden Statusveränderungen an das Framework gesendet. Bei den Statusveränderungen wird zwischen *Inside* (Innerhalb), *Outside* (Außerhalb) und *Unknown* (Unbekannt) unterschieden. Das Framework besitzt weiterhin ein zu delegierendes Objekt, welches bei Statusveränderungen benachrichtigt wird.

4.3 Erfassung mittels Geofencing

Analog zu 4.2 wurde für Geofencing ebenfalls ein Framework umgesetzt. Dieses ähnelt dem Aufbau des iBeacon-Frameworks, da die benötigten Funktionen für das Monitoring identisch sind. Für das Hinzufügen von Geofences wurde ebenfalls ein Protocol implementiert, welchem Objekte die den Methoden startMonitoring(:) und stopMonitoring(:) übergeben werden folgen müssen. Da iOS nur eine begrenzte Anzahl an Geofences zur gleichen Zeit Monitoren kann, besitzt das Protocol eine weite Methode „didFailMonitoring“. Diese wird aufgerufen, falls während des hinzufügen's die zulässige Anzahl an Geofences überschritten wird. Das region-Objekt gibt dabei die optionale Region an, für welche der Fehler aufgetreten ist. Das NSError-Objekt stellt verschiedene Informationen, wie etwa den Fehlercode sowie eine lokalisierte Beschreibung, bereit. Die maximale Anzahl an überwachten Regionen beträgt pro App 20[31].

Wie auch das iBeacon-Framework besitzt das Geofencing-Framework ein delegiertes Objekt, welches bei Statusveränderungen benachrichtigt wird. Nach dem erfolgreichen hinzufügen eines neuen Geofences werden nun beim Betreten und Verlassen die Methoden didEnter(:) und didExit(:) aufgerufen.

[31] `https://developer.apple.com/library/ios/documentation/UserExperience/Conceptual/LocationAwarenessPG/RegionMonitoring/RegionMonitoring.html` [letzter Zugriff: 11.08.2015]

5 Performanceanalyse

5.1 Versuchsaufbau

Da die beiden Verfahren Geofencing und iBeacon unterschiedliche Anwendungsgebiete besitzen, werden diese im nachfolgenden Kapitel getrennt behandelt. Wie in 3.3.1 beschrieben, findet sich die Eignung von iBeacon hauptsächlich im Innenbereich und die Eignung für Geofencing im Außenbereich. Die beiden Verfahren werden nachfolgend hinsichtlich der in 3.2 beschriebenen Anforderungen untersucht.

iBeacon

Die Untersuchung für iBeacon wurde in einem Büro durchgeführt. Als genutzte iBeacons kamen ein iPhone 5 und ein MacBook Pro mit jeweiliger Software zum Einsatz (siehe Abbildung: 5.1). Auf dem iPhone 5 war hierfür die von Apple erstellte Beispiel-App „AirLocate“[32] installiert. Diese wurde für die Nutzung noch leicht modifiziert, da die App für die angepassten Nutzungsrechte von Core Location nicht optimiert war. AirLocate bietet eine einfache Möglichkeit von einem iPhone aus ein iBeacon zu simulieren. Weitere Funktionen dieser App sind das Konfigurieren der Signalstärke verschiedener iBeacons und das Monitoring von iBeacon-Regionen, welche für den Versuch jedoch nicht genutzt wurden. Die Funktionalität des Monitorings wird bereits durch die Prototyp-App (siehe Kapitel: 4) abgedeckt. Auf dem MacBook wurde ein ähnliches Programm genutzt, welches auf GitHub verfügbar ist[33]. In dem Programm „iBeaconSwiftOSX“ ist es möglich, ein iBeacon über ein spezielles User-Interface zu konfigurieren und zu starten. Das Programm wurde ebenfalls modifiziert um einen Zeitstempel beim Ein- sowie Ausschalten auszugeben. In der Prototyp-App (siehe Kapitel: 4) wird beim Eintreffen der Notification ebenfalls ein Zeitstempel generiert. Aus diesen beiden Werten kann somit die Verzögerung berechnet werden, welche sich beim Betreten einer Beacon-Region ergibt. Es fanden nur Messungen der Verzögerung beim Ein- bzw. Ausschalten der Beacons statt. Dies hat den Hintergrund, dass bei einem realen „Betreten“ eines iBeacon-Bereiches keine genaue Aussage getroffen werden kann, wo der Bereich beginnt bzw. welche Ausdehnung dieser einnimmt. Es fand somit eine Evaluierung der Reaktionszeit statt.

Geofencing

Wie in 3.3.2 beschrieben, eignet sich Geofencing vorwiegend für den Außenbereich.

[32] `https://developer.apple.com/library/ios/samplecode/AirLocate/Introduction/Intro.html` [letzter Zugriff: 12.08.2015]

[33] `https://github.com/jmig/iBeaconOSX` [letzter Zugriff: 12.08.2015]

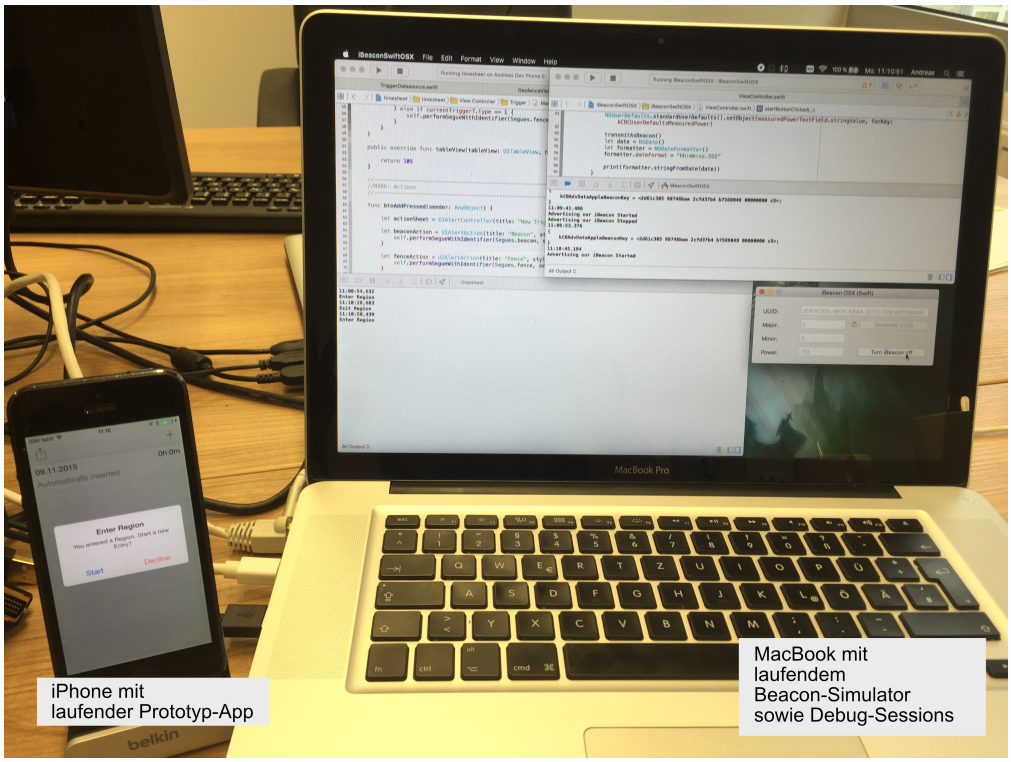


Abbildung 5.1: Foto des iBeacon Versuchsaufbaus. Links das iPhone mit laufender Prototyp-Anwendung, Rechts das MacBook mit laufenden Debug-Sessions für die Prototyp-App und das iBeacon Tool

Der Versuch wurde daher außerhalb eines Gebäudes in einem urbanen Umfeld durchgeführt. Um eine gute Aussage über die Genauigkeit von Geofencing treffen zu können, wurden die Abstände zum äußeren Rand der definierten Region gemessen. Beim Anlegen eines Geofences muss ein Mittelpunkt und ein Radius angegeben werden. Daher errechnet sich der Abstand zur Grenze durch den Abstand zum Mittelpunkt abzüglich des Radius. Für den Versuch wurde der Geofence an verschiedenen Stellen betreten und verlassen (siehe Abbildung: 5.1). Hierbei wurde jeweils der Abstand über die Geokoordinaten des Mittelpunktes und des aktuellen Standortes bei Eintreffen der Benachrichtigung durch die Prototyp-App ermittelt. Weiterhin wurde dieser Versuch wiederholt mit jeweils zwanzig Messungen durchgeführt. Der CLCoreLocationManager besitzt die Eigenschaft „desiredAccuracy“, welche die gewünschte Genauigkeit der Positionsbestimmung reguliert. Während des ersten Versuchs wurde die Genauigkeit auf „NearestTenMeters“ gesetzt, was bedeutet, dass sich die aktuelle Position auf 10 Meter genau ermitteln lässt, sofern ein ausreichendes Signal vorhanden ist. Der zweite Test wurde mit der Genauigkeit „Best“ durchgeführt, welche für diese Zwecke die höchste Genauigkeit darstellt. Die Unterscheidung in diese beiden Genauigkeiten fand statt, da je nach ausgewählter Einstellung der Akku des Mobilgerätes weniger oder mehr belastet wird. CoreLocation bietet noch eine höhere Genauigkeit, „BestForNavigation“, welche allerdings für den Anwendungsfall wenig sinnvoll wäre, da Apple in der Dokummentation selbst empfiehlt diese Genauigkeit nur zu Verwenden, wenn das Gerät an eine Strom-

quelle angeschlossen ist[34].

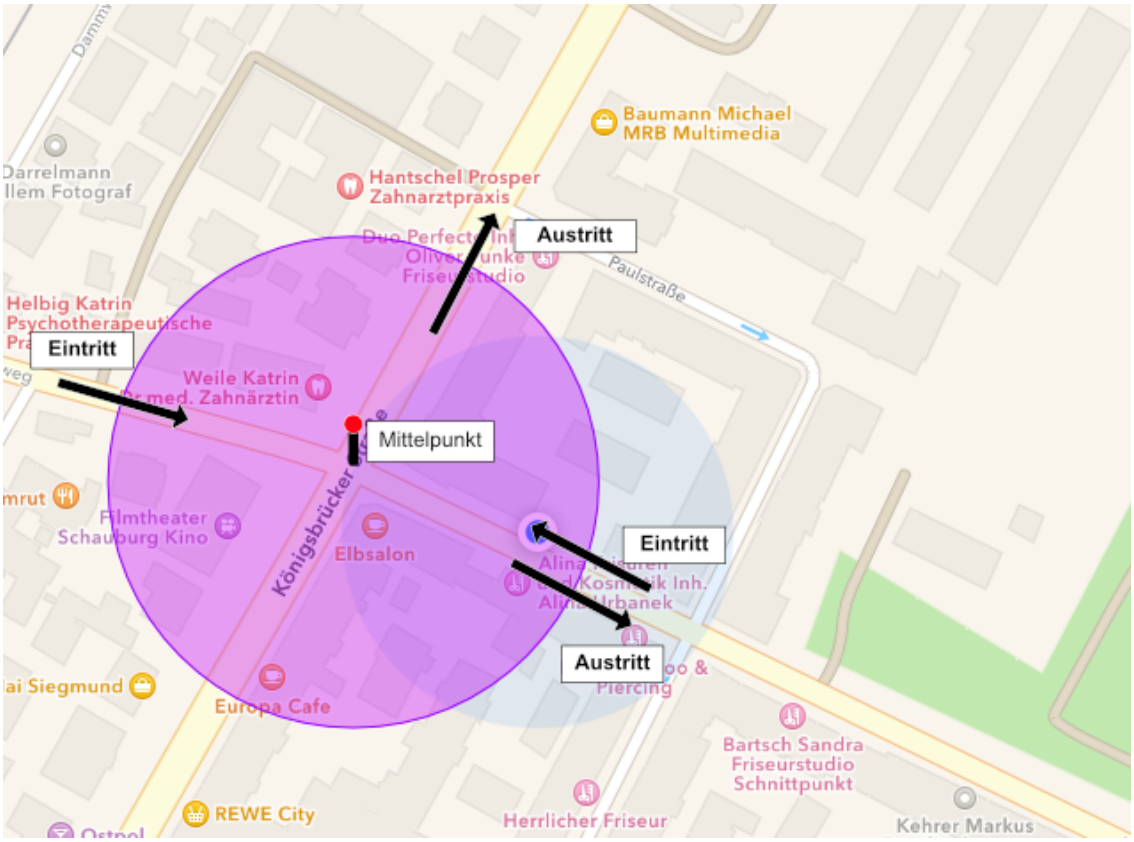


Abbildung 5.2: Darstellung des Geofencing-Versuchs mit verschiedenen Ein- sowie Austrittspunkten

5.2 iBeacon

Um eine relativ genaue Aussage über die Verzögerung treffen zu können, wurden 20 Messwerte erfasst. Jeweils beim Eintritt in die Beacon-Region und beim Austritt aus eben dieser, repräsentiert durch das Ein- sowie Ausschalten des iBeacons. Die erfassten Messewerte sind in Tabelle 5.1 ersichtlich. Die angegebenen Werte wurden aus der zeitlichen Differenz der erfassten Zeitstempel ermittelt.

Die durchschnittliche Verzögerung bei der Registrierung belief sich dabei auf 317 Millisekunden. Beim Verlassen betrug die durchschnittliche Verzögerung 29 Sekunden und 727 Millisekunden. Weiterhin wurden der Wert der Standardabweichung für beide Verzögerungen bestimmt. Die Standardabweichung gibt an, wie stark die Stichproben um

[34] `https://developer.apple.com/library/mac/documentation/CoreLocation/Reference/CoreLocationConstantsRef/#//apple_ref/doc/constant_group/Accuracy_Constants` [letzter Zugriff: 12.10.2015]

Messung	Verzögerung bei Eintritt in s	Verzögerung bei Austritt in s
1	0,023	30,264
2	0,019	29,494
3	0,087	29,068
4	0,765	30,059
5	0,272	30,098
6	0,579	29,203
7	0,023	30,081
8	0,770	30,000
9	0,024	29,074
10	0,019	29,619
11	0,018	28,586
12	0,585	29,807
13	0,955	30,038
14	0,396	29,425
15	0,022	30,196
16	0,019	30,032
17	0,765	29,425
18	0,763	30,130
19	0,025	30,202
20	0,203	29,748
∅	0,317	29,727
σ	0,341	0,472

Tabelle 5.1: Verzögerung der Erkennung des iBeacons durch die Prototyp-App

den Mittelwert herum schwanken. Zu erst wird mittels der Formel:

$$Var(x) = \frac{\sum_{n=1}^{\infty}(x_i - \mu)^2}{n} \tag{5.1}$$

die Varianz Var(X) errechnet. Aus der Varianz lässt sich nun durch Radizieren die Standardabweichung errechnen:

$$\sigma = \sqrt{Var(X)} = \sqrt{\frac{\sum_{n=1}^{\infty}(x_i - \mu)^2}{n}} \tag{5.2}$$

Bezogen auf den Mittelwert traten bei der Registrierung eines iBeacons insgesamt die größten Schwankungen der Verzögerung auf. Die Standardabweichung betrug 341 Millisekunden. Im Hinblick auf die Gesamtheit der Ergebnisse ist die Registrierungsdauer jedoch sehr positiv einzuschätzen, da der Wert von 317 Millisekunden als nicht praxisrelevant einzustufen ist. Die Zeiterfassung würde daher sofort bei betreten des Arbeitsplatzes beginnen. Bezogen auf den Mittelwert von 29,727 Sekunden ist die Standardabweichung von 472 Millisekunden sehr gering. 30 Sekunden Verzögerung sind als relativ hoch einzustufen, liegen allerdings immer noch unter einer Minute wodurch

die Anforderung an die Reaktionszeit erfüllt sind. Für die aktuellen Messwerte kann eine Normalverteilung angenommen werden, da sich die Messwerte annähernd symmetrisch um den Mittelwert herum verteilen (siehe Grafik: 5.3). Dieses Verhalten könnte bestätigt werden wenn für die Messungen weitere Messwerte erfasst würden. Aus den Diagrammen 5.6 geht hervor, dass die Verzögerung beim Eintritt in eine Beacon Region mit einer Wahrscheinlichkeit von 90% unter 750 Millisekunden liegt. Der Wert für den Austritt liegt mit einer Wahrscheinlichkeit von 90% unter 30,3 Sekunden. Aus diesen Werten lässt sich ableiten das beim Eintritt nur zu vernachlässigende Verzögerungen auftreten. Rund 30 Sekunden sind für den Austritt für die meisten Anwendungsfälle vertretbar. Es kann, aufgrund der geringen Standardabweichung, eine Korrektur für jeden Wert vorgenommen werden.

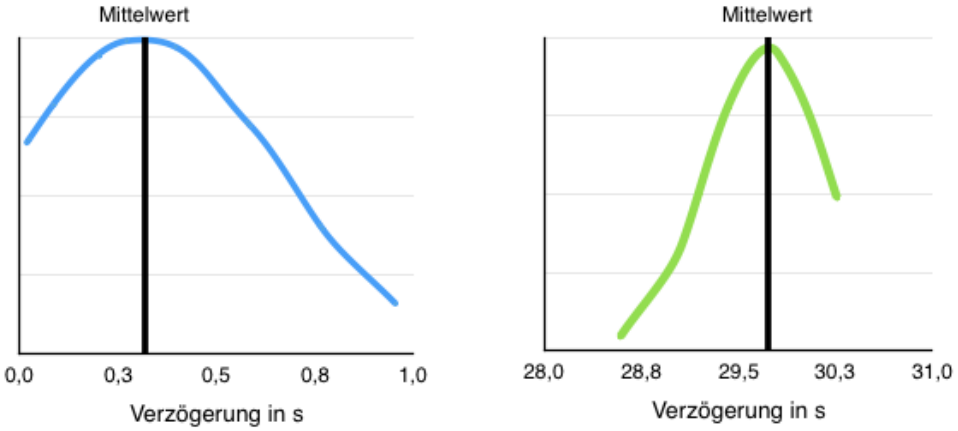


Abbildung 5.3: Darstellung der Normalverteilung des iBeacon-Versuchs

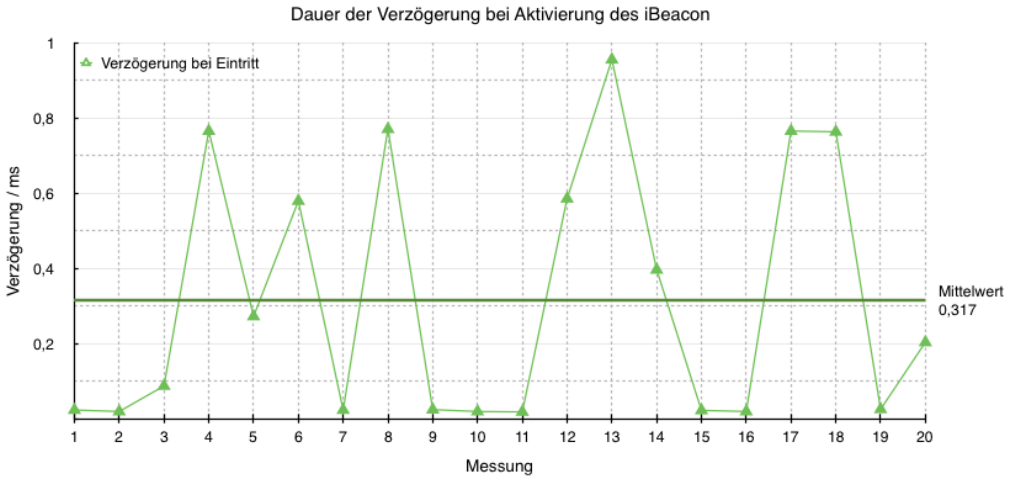


Abbildung 5.4: Messwerte des iBeacon Versuchs

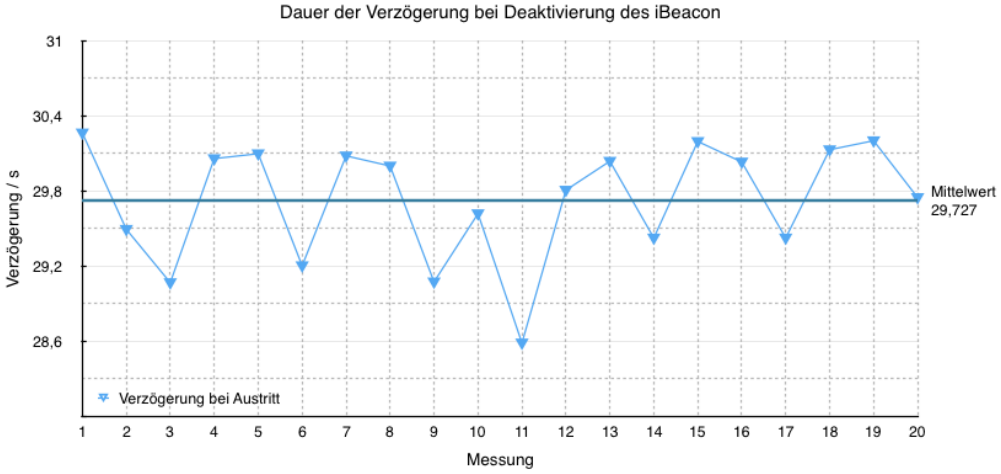
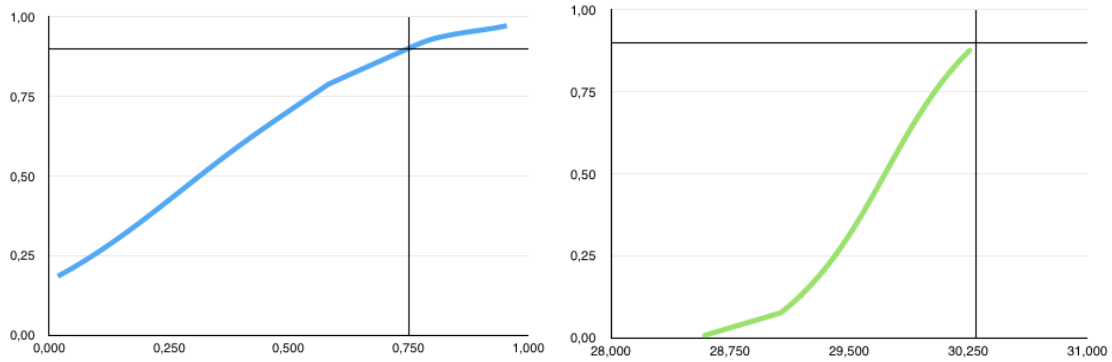


Abbildung 5.5: Messwerte des iBeacon Versuchs

5.3 Geofencing

In [App14b] beschreibt Apple, dass eine Benachrichtigung durch das Betriebssystem nach Betreten oder Verlassen einer Region bis zu 20 Sekunden in Anspruch nehmen kann. Ein Wert von 20 Sekunden würde unter dem in den Anforderungen beschrieben Wert für die Reaktionszeit von einer Minute liegen. Für die Untersuchung der örtlichen Genauigkeit von Geofencing wurden, wie 3.2 beschrieben, der Abstand zum äußeren Rand des Geofences bei Eintreffen der Benachrichtigung des Frameworks gemessen. Die Messwerte wurden dabei für zwei unterschiedliche Genauigkeiten erfasst, welche in Tabelle 5.2 ersichtlich sind. Bei diesem Versuch konnte für die Genauigkeit *CLLocationAccuracyNearestTenMeters* (nachfolgend: NearestTen) eine wesentlich größere Streuung der Messwerte als bei der Genauigkeit *CLLocationAccuracyBest* (nachfol-



(a) Eintritt (b) Austritt

Abbildung 5.6: Darstellung der Kumulativen Verteilung der iBeacon-Messwerte

gend: Best) festgestellt werden. Dies konnte durch die Ermittlung der Varianz sowie der daraus resultierenden Standardabweichung mittels der Formeln 5.1 und 5.2 bestätigt werden.

NearestTen

Für die geringere Genauigkeit NearestTen wurde ein Mittelwert von -6,3m beim Eintritt und 158,6m beim Austritt aus dem Geofence ermittelt. Der negative Wert beim Eintritt bedeutet dabei, dass die Benachrichtigung durch das Framework erst nach dem Betreten des Geofences erfolgte. Die Benachrichtigung für den Austritt hingegen erfolgte immer außerhalb der Region. Dabei war zu beobachten, dass beim Betreten ein wesentlich geringerer Abstand als beim Verlassen auftrat. Die Standardabweichung für das Betreten liegt bei einem Wert von 39,5m, beim Verlassen 68,6m. Aus den Diagrammen 5.9, welche die kumulativen Wahrscheinlichkeiten für den Eintritt sowie Austritt darstellen, geht hervor, dass der Wert für den Eintritt mit einer Wahrscheinlichkeit von 90% weniger als 45 Meter beträgt. Für den Austritt beträgt die Distanz zum Geofence mit einer Wahrscheinlichkeit von 90% weniger als 250 Meter.

Best

Für die Genauigkeit Best wurde ein Mittelwert von 15,2m beim Eintritt und 128,1m beim Austritt errechnet. Diese Werte sind in Bezug auf die geringere Genauigkeit deutlich positiver Einzuschätzen. Die Standardabweichung beträgt beim Eintritt 11,6m und beim Austritt 7,6m. Diese Werte sind deutlich geringer als jene der NearestTen-Messung, was als positiv zu betrachten ist. Die Erwartung, dass sich nach dem Umstellen der Genauigkeit auf Best, die Messwerte enger um den Mittelwert verteilen konnte somit bestätigt werden (siehe Grafik 5.7).

Bei dieser Genauigkeit beträgt die Distanz zum Geofence mit einer Wahrscheinlichkeit von 75% weniger als 23 Meter beim Eintritt und mit 90% Wahrscheinlichkeit weniger als

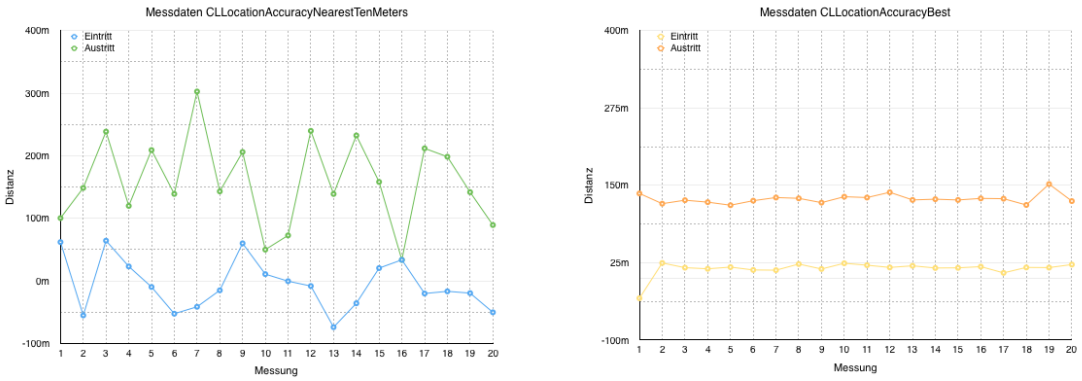
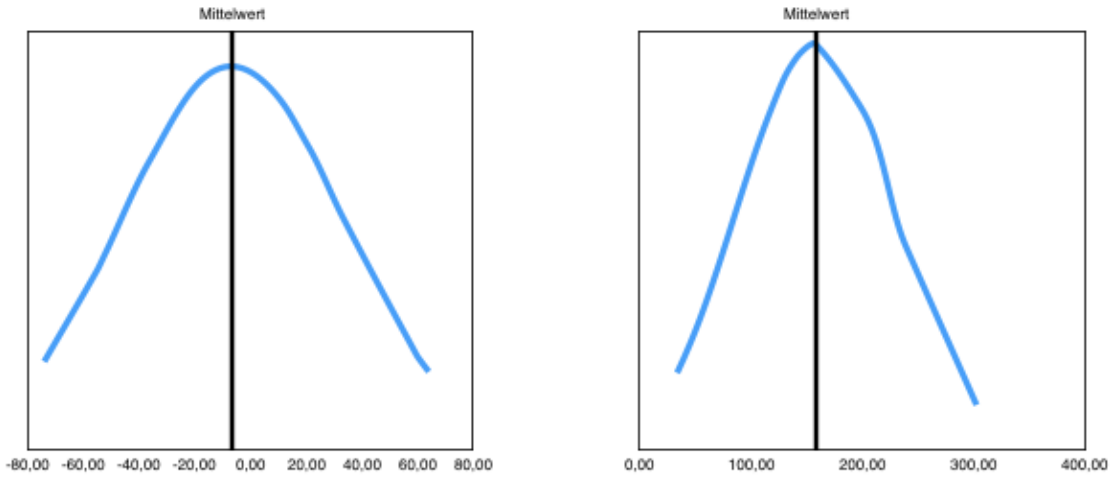


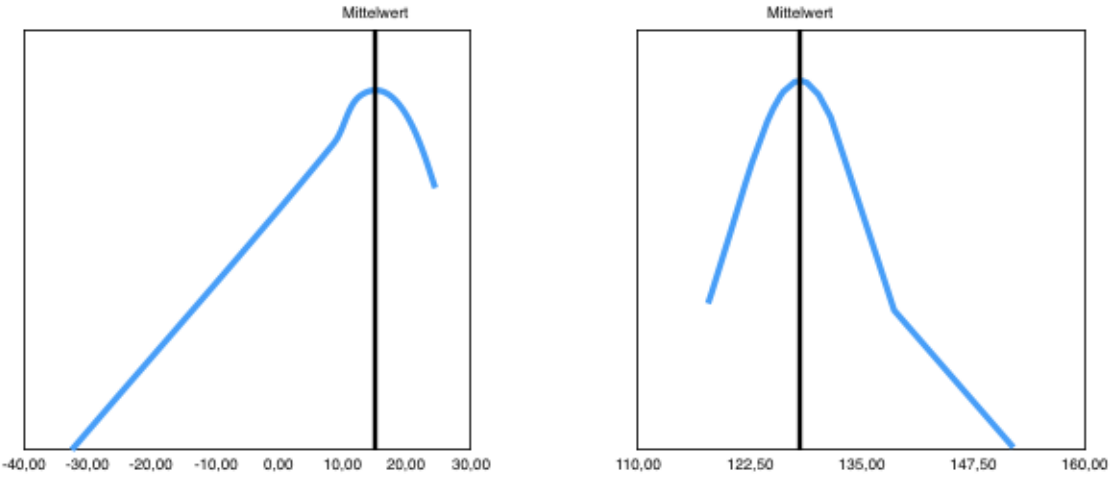
Abbildung 5.7: Vergleich der Messwerte bei unterschiedlichen Genauigkeiten

138 Meter beim Austritt.

Bezogen auf den Radius des für die Messungen verwendeten Geofences von 100m ist der Wert für den Eintritt bei der besseren Genauigkeit als positiv einzustufen. Der Wert für den Austritt hingegen ist bei beiden Genauigkeiten als eher negativ zu betrachten, da bezogen auf 100m Radius 158,6m bzw. 128,1m einen sehr großen Wert darstellen. Um eine genau Aussage über die Genauigkeit treffen zu können, müssten jedoch weitere Versuche mit unterschiedlichen Geofences durchgeführt werden. Die erfassten Werte für beide Messungen können als annähernd Normalverteilt betrachtet werden. Dies ist aus den Diagrammen 5.8 ersichtlich. Es ist zu erwarten, dass weitere Messwerte eine Annäherung der Kurve an eine Glockenkurve zur folge hätten.



(a) Verteilung NearestTenMeters



(b) Verteilung Best

Abbildung 5.8: Darstellung der Verteilungen für beide Genauigkeiten des Geofencing-Versuchs

5.4 Einflussfaktoren

Die Erkennung des Betretens und Verlassens einer Region kann durch verschiedene Störfaktoren beeinflusst werden. Die Störfaktoren werden nachfolgend für die zugrundeliegenden Technologien Bluetooth und GPS beschrieben.

Für GPS bestehen einige äußere Einflussfaktoren, welche zu Messungenauigkeiten führen können. So kann die Positionsbestimmung in Häuserschluchten oder anderen Positionen, an denen der örtliche Horizont eingeschränkt wird, negativ beeinflusst werden. Weiterhin kann das GPS-Signal wie in [Vö13] beschrieben noch durch einige andere Fehlerquellen beeinflusst werden. Eine mögliche Lösung dieses Problems könnte darin bestehen, den gewählten Bereich des Geofences größer zu wählen und anstelle einer automatischen Zeiterfassung das Starten sowie Stoppen der Aufzeichnung teils manuell zu gestalten. Weiterhin wird die Genauigkeit des unter iOS Verwendeten Co-

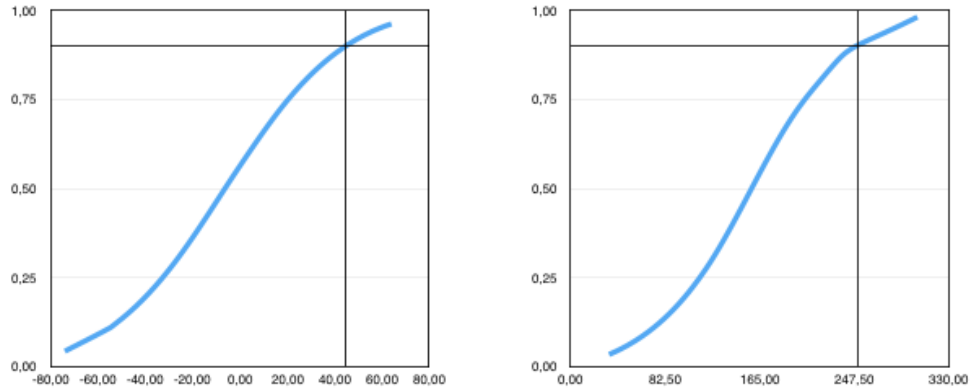


Abbildung 5.9: Darstellung der kumulativen Wahrscheinlichkeiten für die Genauigkeit Nearest-Ten bei Eintritt (links) und Austritt (rechts)

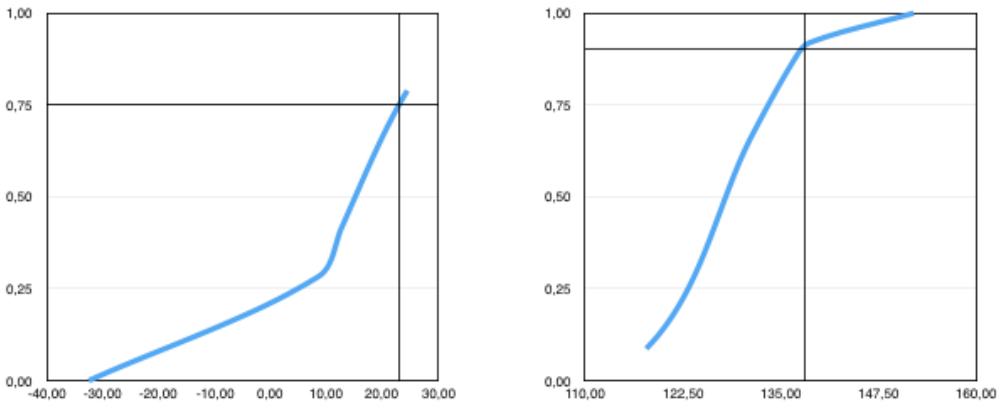


Abbildung 5.10: Darstellung der kumulativen Wahrscheinlichkeiten für die Genauigkeit Best bei Eintritt (links) und Austritt (rechts)

	NerestTen		Best	
	Abstand Eintritt in m	Abstand Eintritt in m	Abstand Eintritt in m	Abstand Eintritt in m
1	61,90	100,29	−32,51	136,80
2	−55,20	148,52	24,47	120,30
3	64,19	238,52	16,89	125,91
4	23,04	119,78	15,06	122,74
5	−9,80	209,00	17,79	117,82
6	−52,54	138,90	13,20	125,11
7	−41,48	302,60	12,86	130,20
8	−15,13	142,97	22,73	128,92
9	59,95	205,93	14,76	121,97
10	10,71	49,89	23,98	131,53
11	−0,53	72,63	20,88	130,19
12	−8,28	239,71	17,27	138,65
13	−74,05	138,83	19,87	126,34
14	−35,86	232,38	16,37	127,53
15	20,38	158,10	16,62	126,28
16	33,53	33,50	18,54	128,79
17	−20,26	211,72	8,49	128,33
18	−16,75	198,33	17,26	118,21
19	−19,63	141,70	17,01	152,00
20	−50,30	89,18	21,91	124,46
∅	−6,31	158,62	15,17	128,10
σ	39,52	68,64	11,59	7,56

Tabelle 5.2: Distanz-Messwerte des Geofencing Versuchs

re Location Frameworks stark durch die vom Entwickler eingestellte „desired Accuracy“ beeinflusst. Aus den Distanzmessungen in 5.3 wird ersichtlich, dass die Genauigkeit zwischen den Stufen stark unterschiedlich ist. Da Apple zur Verbesserung der Genauigkeit des CoreLocation Frameworks zusätzlich zum GPS-Signal noch die Positionen von Mobilfunkmasten und WLAN-Hotspots in die Standortbestimmung einbezieht, können in ländlichen Regionen ebenfalls Ungenauigkeiten auftreten, da sich die Netzabdeckung und die Verteilung von WLAN-Hotspots im Gegensatz zu Städten stark verringert. Auf die Ausdehnung eines Geofences bezogen dürften die Abweichungen jedoch als zu vernachlässigend eingestuft werden.

Ein nicht zu vernachlässigender Faktor ist jedoch eine von Apple implementierte Funktion, welche das schnelle Wechseln des Status (Innerhalb oder Außerhalb) verhindern soll. Würde sich ein Nutzer in unmittelbarer Nähe des äußeren Randes eines Geofences oder der Reichweite eines iBeacons befinden, könnte dies dazu führen, dass schnell hintereinander die jeweiligen Events für Ein- und Austritt ausgelöst werden. Um diese

Verhalten zu unterbinden gibt es eine gewisse Verzögerung bis die jeweiligen Events ausgelöst werden. Zu beobachten ist dieses Verhalten wenn man sich die Distanz zu einem iBeacon von iOS anzeigen lässt. Nach dem ein iBeacon Bereich verlassen wird liefert das Core Location-Framework noch einige Distanzwerte, welche mit -1,0m angegeben sind. Weiterhin wird der Status Unknown zurückgeliefert, was bedeutet, dass das Beacon vom Framework nicht mehr gefunden wird.

Der hauptsächliche Störfaktor der Bluetooth-Technologie ist die unter 3.3.1 bereits beschriebene Dämpfung durch Gegenstände und Personen. Dieser Störfaktor ist jedoch nur für eine genaue Abstandsmessung problematisch. Für eine Präsenzdetektion dürfte das Problem nicht praxisrelevant sein.

5.5 Auswertung

Anhand der erfassten Messwerte des iBeacon-Versuchs lässt sich schlussfolgern, dass eine mobile Arbeitszeiterfassung mittels eines Smartphones und an den Arbeitsplätzen befindlicher iBeacons gut möglich ist. Die gemessenen Reaktionszeiten liegen unter den in 3.2 aufgestellten Anforderungen. Die beim Eintritt in die Beacon-Region aufgetretene durchschnittliche Verzögerung kann - aufgrund des relativ geringen Wertes von 317 Millisekunden - vernachlässigt werden. Die Verzögerung bei Austritt von durchschnittlich 29,727 Sekunden kann aufgrund der geringen Standardabweichung von der Anwendung ausgeglichen werden, in dem der Wert von der Arbeitszeit abgezogen wird. Im Hinblick auf den Energiebedarf kann davon ausgegangen werden, dass das reine Überwachen einer Beacon-Region die Akku-Laufzeit eines Gerätes nur wenig beeinträchtigt, jedoch kann die Akkulaufzeit durch die Menge der zu überwachenden Regionen beeinträchtigt werden [ais14]. Wie in [CDMN+14] beschrieben, ist der Energiebedarf im Gegensatz zur Übermittlung der erfassten Daten, an beispielsweise einen Webserver, wesentlich geringer. Weiterhin konnte während der Messung der Reaktionszeiten kein erhöhter Energiebedarf der Prototyp-Anwendung festgestellt werden. Das iBeacon-Verfahren eignet sich daher sowohl für ein Benachrichtigungs-basiertes, sowie für ein automatisches Erfassen von Arbeitszeiten innerhalb von Gebäuden.

Die für Geofencing erfassten Messungen legen nahe, dass eine Zeiterfassung mit der eingestellten Genauigkeit *CLLocationAccuracyNearestTenMeters* nur sehr ungenaue Ergebnisse liefern dürfte. Im Gegensatz zur Messung mit der Genauigkeit *CLLocationAccuracyBest* streuen die Messwerte sehr stark um den Mittelwert. Die in 3.2 definierten Anforderungen zur örtlichen Genauigkeit konnten daher nicht eingehalten werden. Wie in Diagramm 5.7 ersichtlich ist, ist die Streuung der Messwerte bei der Genauigkeit Best wesentlich geringer. Somit lässt sich einfacher eine Aussage über die zu erwartende Entfernung treffen. Da der Abstand jedoch bei beiden Messungen zu erheblich war, kann von einer automatischen Zeiterfassung mittels Geofencing abgeraten werden. Hierbei sollte auf ein Benachrichtigungs-basiertes Verfahren zurückgegriffen werden, bei welchem der Nutzer eine Push-Mitteilung erhält, wenn die Zeiterfassung starten oder stoppen kann. Dieses System wäre dann mit Apples „Erinnerungen“-App vergleichbar. Um eine genauere Aussage über die Genauigkeit treffen zu können, müssen noch weitere Messungen mit unterschiedlichen Geofences durchgeführt werden. Bei beispielsweise einem weitläufigeren Firmengelände könnte der Abstand noch als vertretbar eingestuft werden.

6 Fazit und Ausblick

Ziel der vorliegenden Arbeit war es, die Eignung eines Smartphones, speziell die eines iPhones, für das Einsatzgebiet der standortbasierten Zeiterfassung zu evaluieren. Hierbei wurden die beiden Verfahren Geofencing und iBeacon betrachtet.

Für den einfachen Anwendungsfall einer benachrichtigungsbasierten Zeiterfassung, bei welcher der Nutzer mittels einer Nachricht auf seinem Smartphone an das Erfassen seiner Arbeitszeit erinnert wird, eignen sich die beiden Verfahren, iBeacon und Geofencing, hinreichend gut. Der Nutzer kann dabei selbst entscheiden, ab oder bis wann die abzurechnende Arbeitszeit erfasst wird.

Das iBeacon-Verfahren lässt sich für die standortbasierte Zeiterfassung gut konfigurieren, da eine relativ genaue Abstandsmessung vom Nutzer, also dessen Smartphone, zum Beacon vorgenommen werden kann. Die Reaktionszeit, die ein Smartphone - im konkreten Fall ein iPhone 5 - benötigt, bis erkannt wird, dass sich ein spezielles Beacon in der Nähe befindet, bewegte sich während der vorgenommenen Messungen in einem für eine Zeiterfassung akzeptablen Bereich.

Die Analyse der Distanzmessungen des Geofencing-Versuchs hat ergeben, dass die örtliche Genauigkeit stark von der durch den Programmierer eingestellten „desiredAccuracy“ abhängt. Zwar kann anhand der Messwerte der höheren Genauigkeit „CLLocationAccuracyBest“ darauf geschlossen werden, dass sich die Entfernungen mit hoher Wahrscheinlichkeit innerhalb eines bestimmten Bereiches bewegen, doch gilt dies nur für den im Versuchsaufbau beschriebenen Geofence. Um eine allgemeinere Aussage über die örtliche Genauigkeit treffen zu können, sind jedoch weitere Versuche mit unterschiedlichen Geofences, dass heißt unterschiedlichen Radien, Positionen sowie Eintritts- sowie Austrittsgeschwindigkeiten durchzuführen. Insgesamt kann festgestellt werden, dass Aufgrund der Vielfältigkeit eines Geofences die Frage nach der Eignung für ein mobiles Zeit-Tracking differenziert betrachtet werden muss. Sollten sich die Messwerte für andere Geofences in einem ähnlichem Bereich bewegen, könnte ein System implementiert werden, welches basierend auf der Geschwindigkeit und des eventuell wichtigen Radius des Geofences die Arbeitszeit automatisiert erfasst und berechnet. Weiterhin konnte während der Messungen zwar kein erheblich höherer Energiebedarf festgestellt werden, was nicht garantiert, dass sich dieser während einer Produktivnutzung ebenso verhält. Es ist daher noch zu evaluieren, wie sich der Energiebedarf einer solchen Anwendung mit hinreichender Standortgenauigkeit bei einer täglichen Produktivnutzung und dem mehrfachen Ein- und Austritt in bzw. aus Geofences verhält.

Es bietet sich in aufbauenden Arbeiten an, die Performance auch auf weiteren Plattformen, wie Android und Windows Phone, zu evaluieren. Als Alternative zur kooperativen Eigenortung, könnte die Betrachtung zudem auf Verfahren, die auf kooperativer Frem-

dortung basieren, ausgeweitet werden.

Die beiden im Rahmen dieser Arbeit entstanden Softwarekomponenten müssen für einen Produktivgebrauch und das Einbinden in weitere Projekte noch leicht modifiziert werden. So muss noch eine korrekte Fehlerbehandlung für die in *Core Location* möglichen Fehler hinzugefügt werden. Weiterhin kann es sinnvoll sein, die beiden Frameworks zu einem einzigen zu kombinieren, da sich die Strukturen und die ausgelösten Ereignisse sehr ähneln. Sinnvoll wäre auch die Frameworks für die Verwendung mit *CocoaPods* zu optimieren, da so der Aufwand des Einbindens in weitere Projekte verringert würde.

Anhang A: Verwendete Software

A.1 Programmierung

Für die Programmierung der Prototyp-App und der Erstellung der Frameworks kam die Entwicklungsumgebung Xcode in Version 7.1 von Apple zum Einsatz. Die App sowie die beiden Frameworks wurden in der Programmiersprache Swift 2 umgesetzt und sind auf iPhones mit dem Betriebssystem iOS 9 lauffähig. Getestet wurde die Anwendung auf einem iPhone 6 und verschiedenen Emulatoren.

A.2 Bachelordokument

Das vorliegende Dokument wurde mittels der Textverarbeitung LaTex erstellt. Die zugrunde liegende Dokumentklasse stammt von der Hochschule Mittweida und wurde durch Herr Prof. Klaus Dohmen erstellt[35]. Die Klasse wurde noch geringfügig modifiziert. Die in dieser Arbeit abgebildeten Diagramme wurden mit der Software Numbers in Version 3.6.1 erstellt. Die Bearbeitung der Bilder fand mit der Software *Affinity Designer* statt.

[35] `https://www.cb.hs-mittweida.de/webs/dohmen/service/vorlagen.html`

Literaturverzeichnis

[ais14] AISLELABS: *iBeacon Battery Drain on Apple vs Android*. http://www.aislelabs.com/reports/ibeacon-battery-drain-iphones/. Version: August 2014. – [letzter Zugriff: 16.09.2015]

[App14a] APPLE: *Getting Started with iBeacon*. https://developer.apple.com/ibeacon/Getting-Started-with-iBeacon.pdf. Version: Juni 2014. – [letzter Zugriff: 14.10.2015]

[App14b] APPLE: *Location and Maps Programming Guide*. https://developer.apple.com/library/ios/documentation/UserExperience/Conceptual/LocationAwarenessPG/LocationAwarenessPG.pdf. Version: Oktober 2014. – [letzter Zugriff: 11.08.2015]

[CDMN+14] CONTE, Giorgio ; DE MARCHI, Massimo ; NACCI, Alessandro A. ; RANA, Vincenzo ; SCIUTO, Donatella: BlueSentinel: a first approach using iBeacon for an energy efficient occupancy detection system. In: *1st ACM International Conference on Embedded Systems For Energy-Efficient Buildings (BuildSys)*, 2014

[HB14] HACHIYA, Tatsuro ; BANDAI, Masaki: SmartLocService: Place Identification Method Using Space Dependent Information for Indoor Location-Based Services. In: *Advanced Information Networking and Applications Workshops (WAINA), 2014 28th International Conference on* IEEE, 2014, S. 578–581

[Hic09] HICKMANN, Robert: *Entwicklung eines mobilen Messsystems zur Positionsbestimmung und Signalstärkemessung in Wireless-LAN-Netzwerken*, Diss., 2009

[HZH+12] HOFLINGER, Fabian ; ZHANG, Rui ; HOPPE, Jens ; BANNOURA, Amir ; REINDL, Leonhard M. ; WENDEBERG, Johannes ; BUHRER, M ; SCHINDELHAUER, Christian: Acoustic self-calibrating system for indoor smartphone tracking (assist). In: *Indoor Positioning and Indoor Navigation (IPIN), 2012 International Conference on* IEEE, 2012, S. 1–9

[IDC15] IDC, International Data C.: *Worldwide Smartphone Growth*. http://www.idc.com/getdoc.jsp?containerId=prUS25860315. Version: August 2015. – [letzter Zugriff: 16.09.2015]

[LR10] LANGER, Josef ; ROLAND, Michael: *Anwendungen und Tech-*

nik von Near Field Communication (NFC). Berlin, Heidelberg : Springer-Verlag Berlin Heidelberg, 2010. http://dx.doi.org/10.1007/978-3-642-05497-6. http://dx.doi.org/10.1007/978-3-642-05497-6. – ISBN 978–3–642–05496–9

[Mah98] MAHAFZA, Bassem R.: *Introduction to radar analysis*. CRC press, 1998

[Mas93] MASUMOTO, Yutaka: *Global positioning system*. Mai 11 1993. – US Patent 5,210,540

[Mat11] MATT, Christian: Lösungen für mobile Arbeitszeiterfassung. In: *Zeitschrift für Controlling & Management* (2011), Nr. 4, S. 201–205

[RTI13] RAJ, CP R. ; TOLETY, SeshuBabu ; IMMACULATE, Catheine: QR code based navigation system for closed building using smart phones. In: *Automation, Computing, Communication, Control and Compressed Sensing (iMac4s), 2013 International Multi-Conference on* IEEE, 2013, S. 641–644

[TT10] TAMM, Gerrit ; TRIBOWSKI, Christoph: *RFID*. Springer-Verlag, 2010

[Vö13] VÖLKER, Johannes: *Energie-effizientes Tracking von Personen mit dem Smartphone*, Diplomarbeit, Technische Universität Dresden, Diss., 2013

Glossar

Apple Pay	Ein auf iPhone und Apple Watch bereitgestellter Dienst zum kontaktlosen Bezahlen.
Framework	Ein Programmiergerüst welches keine eigene Anwendung darstellt, eine existierende Anwendung jedoch um Funktionen ergänzt.
Galileo	Ein im Aufbau befindliches System zur globalen Satellitengestützten Navigation.
GitHub	Ein Web-basierter Dienst zum Speichern und Verwalten von meist Softwareprojekten.
GSM	Ein Standard für volldigitale Mobilfunknetze welche hauptsächlich für Telefonie genutzt werden.
Latitude	Englisch für die geographische Breite, auch Breitengrad.
Longitude	Englisch für die geographische Länge, auch Längengrad.
Mac OS	Betriebssystem für Laptops und Desktop- Rechner der Firma Apple.
MIT-Lizenz	Eine Software-Lizenz welche die unentgeltliche Nutzung, Verbreitung, Verwendung, Änderung sowie Unterlizensierung der unter ihr stehenden Software erlaubt.
nuget	Ein in der Programmierumgebung Visual Studio integrierter Paketmanager.
Persistierung	Sichern von Daten auf einem nicht-flüchtigem Speichermedium.
Proximity	Englisch für Nähe, Näherung.
SDK	Unter einem Software Developement Kit, kurz SDK, wird eine Sammlung Werkzeugen verstanden welche für das erstellen von Anwendungen verwendet werden kann.

Universal Windows Platform	Anwendungs-Architektur welche das Entwickeln für Windows-Tablets, Smartphones und Desktop-Rechner ermöglicht.
URL	Ein Uniform Resource Locator wird zum identifizieren und lokalisieren einer Ressource, zum Beispiel eine Internet-Seite, in einem Rechnernetz verwendet.

Erklärung

Hiermit erkläre ich, dass ich meine Arbeit selbstständig verfasst, keine anderen als die angegebenen Quellen und Hilfsmittel benutzt und die Arbeit noch nicht anderweitig für Prüfungszwecke vorgelegt habe.

Stellen, die wörtlich oder sinngemäß aus Quellen entnommen wurden, sind als solche kenntlich gemacht.

Mittweida, 20.11.2015